École doctorale n° 401 : Sciences Sociales

THÈSE

pour obtenir le grade de docteur délivré par

Université Paris 8

Spécialité doctorale "Informatique"

présentée et soutenue publiquement par

Ngoc Nguyen Thinh DONG

le 4 Juillet 2017

Révision d'ontologies fondée sur tableaux

Directrice de thèse : **Myriam LAMOLLE**
Co-encadrant de thèse : **Chan LE DUC**

Jury

Olivier Curé,	MCF-HDR, Université Paris Est-Créteil, Marne-la-Vallée	Rapporteur
Jérôme Euzénat,	Directeur de recherche, HDR, INRIA	Rapporteur
Arab Ali Chérif,	Professeur, Université Paris 8	Examinateur
Odile Papini,	Professeur, Université d'Aix-Marseille	Examinateur

Université Paris 8
Laboratoire d'informatique avancée de Saint-Denis (LIASD)
IUT de Montreuil
Saint-Denis, France

THESE

Résumé

L'objectif de cette thèse est d'étendre des opérateurs de révision d'ontologie existants en respectant les postulats *AGM* (priorité aux nouvelles connaissances, cohérence de connaissances et minimalité des changements) pour des ontologies d'expressivité $\mathcal{SHIQ}$ et de proposer de nouveaux algorithmes palliant les inconvénients inhérents à ces opérateurs.

Après étude de l'existant, nous avons proposé un nouvel algorithme de tableau pour la révision des ontologies exprimées en $\mathcal{SHIQ}$. En créant ce nouvel algorithme de tableau, nous avons défini la notion des *modèles de graphe* finis (des *modèles d'arbre* ou des *modèles de forêt*) afin de représenter un ensemble éventuellement infini de modèles d'une ontologie en $\mathcal{SHIQ}$. Ces structures finies équipées d'un pré-ordre total permettent de déterminer la différence sémantique entre deux ontologies représentées comme deux ensembles de modèles.

Nous avons mis en œuvre les algorithmes proposés dans notre moteur de révision OntoRev, intégrant des techniques d'optimisation pour (i) réduire des non-déterminismes lors de l'application de l'algorithme de tableau, (ii) optimiser le temps du calcul de distance entre des modèles d'arbre ou entre des modèles de forêt, (iii) éviter de construire des forêts ou des arbres non nécessaires à la révision. De plus, nous avons examiné la possibilité d'améliorer la méthode de tableau par une approche permettant de compresser les modèles d'arbres. Enfin, nous avons effectué des expérimentations avec des ontologies du monde réel qui ont mis en exergue la difficulté à traiter des axiomes non déterministes intrinsèques.

Nos futurs travaux aborderont (i) l'amélioration de l'implémentation en appliquant des techniques d'optimisation existantes dans l'algorithme de tableau standard, (ii) l'extension de l'algorithme pour la révision des ontologies exprimées dans des logiques plus expressives, *e.g.* $\mathcal{SHIQ}$, (iii) l'application de notre algorithme à la révision d'un réseau d'ontologies alignées.

Mots-clés : Ontologie, Révision, Algorithme de Tableau, Logiques de Description

Abstract

The objective of this PhD thesis is to extend existing ontology revision operators in accordance with the postulates *AGM* (priority on new knowledge, knowledge coherence and minimal change) for ontologies in $\mathcal{SHIQ}$ and propose new algorithms to overcome the disadvantages in these operators.

After studying the existing approaches, we have proposed a new tableau algorithm for the revision of ontologies expressed in $\mathcal{SHIQ}$. Together with this new tableau algorithm, we have defined the notion of finite *graph models* (*tree models* or *forest models*) in order to represent a possibly infinite set of models of an ontology in $\mathcal{SHIQ}$. These finite structures equipped with a total pre-order make it possible to determine the semantic difference between two ontologies represented as two sets of models.

We have implemented the proposed algorithms in our revision engine OntoRev, by integrating optimization techniques for (i) reducing non-determinisms when applying the tableau algorithm, (ii) optimizing the computation time of the distance between tree models or between forest models, (iii) avoiding the construction of unnecessary forests or trees in the revision. In addition, we examined the possibility of improving the tableau method using an approach for compressing tree models. Finally, we carried out experiments with real-world ontologies which highlighted the difficulty to deal with intrinsic non-deterministic axioms.

Our future work will approach to (i) improving the implementation by applying existing optimization techniques in the standard tableau algorithm, (ii) extending the algorithm for the revision of ontologies expressed in more expressive logics, *e.g.* $\mathcal{SHIQ}$, (iii) applying our algorithm to the revision of a network of aligned ontologies.

Keywords : Ontology, Revision, Tableau Algorithm, Description Logics

Remerciements

J'aimerais tout d'abord remercier ma directrice de thèse, Professeure Myriam LAMOLLE, pour m'avoir accueilli au sein de son équipe avec un contrat doctoral ainsi qu'un contrat supplémentaire. Un grand Merci pour sa relecture finale méticuleuse de chacun des chapitres de cette thèse.

J'adresse aussi mes chaleureux remerciements à mon co-encadrant de thèse, M. Chan LE DUC, pour son attention de tout instant sur mes travaux, ainsi que pour ses compétences pédagogiques et scientifiques, ses conseils avisés, sa franchise et sa sympathie. J'ai beaucoup appris à ses côtés et je lui adresse ma gratitude pour tout cela.

Je tiens à remercier M. Olivier CURÉ, Maître de conférences - HDR à l'Université Paris Est-Créteil, Marne-la-Vallée, et M. Jérôme EUZENAT, Directeur de recherche - HDR à INRIA Grenoble, qui m'ont fait l'honneur d'accepter d'être rapporteur de cette thèse.

J'associe à ces remerciements Madame Odile PAPINI, Professeure à l'Université d'Aix-Marseille, et M. Arab ALI CHÉRIF, Professeur à l'Université Paris 8, pour avoir accepté d'examiner mon travail.

Je désire en outre remercier tous mes collègues et amis de l'IUT de Montreuil et de l'Université Paris 8 qui ont contribué au maintien d'une bonne humeur et leur sympathie, à M. Gaétan DARQUIÉ pour m'avoir accueilli chaleureusement au sein du projet PRÉVU.

Ces remerciements seraient incomplets si je n'en adressais pas à ma famille, ma femme et notre fille dont l'amour, l'encouragement constants et le soutien m'ont été d'un très grand réconfort.

Publications associées à cette thèse

Articles de revues internationales

1. Thinh Dong, Chan Le Duc, Myriam Lamolle. Tableau-based Revision for Expressive Description Logics with Individuals. *Journal of Web Semantics*, 2017 (soumis).

Articles de revues nationales

2. Azziz Anghour, Myriam Lamolle, Thinh Dong, Chan Le Duc, Guylain Delmas. Apprentissage de gestes techniques par une architecture multimedia ontologique, *Revue des Nouvelles Technologies de l'Information, Numéro spécial "Conception des Architectures Logicielles"*, version étendue de CAL 2015 et MODA 2015, vol. RNTI-L-8, pages 119-134, 2016.

Articles de conférences internationales

3. Thinh Dong, Chan Le Duc, Myriam Lamolle. Tableau-based Revision for Expressive Description Logics. *Intelligence Artificielle Fondamentale (IAF)*, 2017 (soumis).
4. Thinh Dong, Chan Le Duc, Philippe Bonnot, Myriam Lamolle. Tableau-Based Revision over $\mathcal{SHIQ}$ TBoxes. *Logic for Programming, Artificial Intelligence, and Reasoning - 20th International Conference, LPAR-20, Suva, Fiji, November 24-28, Proceedings*, pages 575-590, 2015.
5. Thinh Dong, Chan Le Duc, Philippe Bonnot, Myriam Lamolle. Tableau-based revision in $\mathcal{SHIQ}$. *Proceedings of the 28th International Workshop on Description Logics, Athens,Greece*, 2015.

Table des matières

Liste des figures

Liste des tableaux

Introduction

Le monde numérique actuel manipule des millions de données par jour ; mais pour pouvoir appréhender l'information et donc la connaissance sous-jacente, il est nécessaire de disposer de modèles de représentation évolués, porteurs de sens en soi, facilitant l'interopérabilité entre systèmes (communication entre machines et/ou entre machines et humains). De nouveaux modèles dits ontologiques ont fait une percée remarquée ces dernières décennies. Dans l'ingénierie des connaissances, une ontologie est une spécification formelle et explicite d'une conceptualisation partagée d'un domaine. L'objectif premier d'une ontologie est de modéliser un ensemble de connaissances dans un domaine donné, qui peut être réel ou imaginaire. Les ontologies ont été appliquées dans un large éventail de domaines pratiques tels que l'e-Science, l'e-Commerce, l'informatique médicale, la bio-informatique, le Web sémantique, etc.

Les applications fondées sur la sémantique encapsulent souvent un ensemble d'ontologies qui représentent les connaissances impliquées dans différentes sources de données. Certaines de ces ontologies évoluent constamment car, non seulement les données sont mises à jour, mais aussi l'environnement des applications et les besoins des utilisateurs sont modifiés au cours du temps. Dans le contexte du projet *LearningCafé*[1], l'objet principal est la mise en place d'une plateforme collaborative capable de proposer à un utilisateur un parcours pédagogique (une séquence de vidéos ou de documents) à suivre pour l'apprentissage d'un métier manuel. La construction d'un tel parcours nécessite des concepts tels que PedagogicalResource, ObjectLearning, Competence, UserLevel, etc. définis dans les ontologies des formations *Training* et des utilisateurs *UserProfile*. Lorsque le niveau d'un utilisateur

1. financement FUI-15

évolue après certaines étapes d'apprentissage, les ontologies devraient être modifiées pour que la plateforme puisse proposer un nouveau parcours pédagogique convenant mieux à son niveau. Cette tâche pourrait entraîner une révision d'une des ontologies simple (voire de tout le réseau) de telle sorte que l'ontologie résultante devrait prendre en compte la nouvelle connaissance.

La révision d'ontologies est un processus dans lequel des axiomes sont automatiquement changés pour refléter l'acquisition de nouvelles informations. Si une ontologie est considérée comme un ensemble d'axiomes, l'abandon ou l'acceptation d'un nouvel axiome correspond aux opérations de contraction ou de révision, respectivement, sur cette ontologie. Hormis la contrainte de succès, l'exécution de ces opérations doit subir d'autres contraintes telles que l'ontologie résultante devrait :

i. être toujours cohérente lorsqu'une nouvelle information est ajoutée,
ii. être représentable par le langage utilisé,
iii. changer le moins possible.

Le problème de la révision des ontologies exprimées en OWL-DL (appelées ontologies en LD) est étroitement lié au problème de la révision de croyances qui a été largement discuté dans la littérature. Parmi les premiers travaux sur la révision de croyances, Alchourrón, Gärdenfors et Makinson (AGM) [Alchourrón *et al.*, 1985] ont introduit des contraintes intuitives et plausibles (appelées postulats d'AGM) qui devraient être satisfaites par tous les opérateurs de révision de croyances.

Cependant, cette approche sémantique ne peut pas être directement employée pour des ontologies en LD. En effet, d'une part, les problèmes essentiels issus de la révision d'une base de connaissance générale sont au-delà du cadre de la logique du premier ordre. D'autre part, certains de ces problèmes deviennent évidents pour les ontologies qui ne contiennent pas d'individu, appelées *terminologies*. Par exemple, lorsqu'une règle est ajoutée à une base de règles existante, il faut assurer que la base modifiée est toujours cohérente. Par contre, ajouter un concept avec sa définition à une terminologie ne cause jamais d'incohérence (sous l'hypothèse que la terminologie accepte un concept insatisfiable). Ceci sous-entend que la terminologie serait protégée contre ce type de modification.

Afin de sensibiliser le lecteur à la difficulté des problèmes liés à la révision d'une ontologie, considérons une ontologie comportant les axiomes suivants :

α : Tous les informaticiens savent programmer.

β : Notre ami est informaticien.

γ : Notre ami est diplômé de l'école Y.

δ : L'école Y forme les informaticiens.

Si un moteur d'inférence a cette ontologie en entrée de son processus de calcul, il peut déduire des axiomes $\alpha - \delta$ que

ϵ : Notre ami sait programmer.

Mais, un jour, nous découvrons que notre ami ne sait pas programmer. Cela nous oblige à ajouter $\neg\epsilon$ à l'ontologie. De ce fait, notre ontologie devient incohérente. Si nous voulons maintenir la cohérence de l'ontologie, cela nécessite une révision. C'est-à-dire qu'un certain nombre d'axiomes doivent être enlevés. Il est évident que nous ne souhaitons pas abandonner tous les axiomes car certains axiomes sont encore pertinents. Donc, nous devons choisir quelques axiomes parmi α - δ à retirer. Le problème essentiel de la révision d'une ontologie est que la seule considération logique ne nous permet pas de déterminer le(-s) axiome(-s) à enlever.

De plus, le fait que les axiomes d'une ontologie comportent également des conséquences logiques complique davantage les choses. Lorsque nous abandonnons un axiome, nous devons décider quelles conséquences sont à ajouter ou à retirer. Par exemple, α a deux conséquences à savoir :

α' : Tous les informaticiens savent programmer sauf notre ami (on ne sait pas si notre ami sait programmer).

α'' : Tous les informaticiens savent programmer sauf certaines personnes qui sont diplômées de l'école Y (on ne sait pas si ces personnes savent programmer).

Si nous décidons d'enlever α pour prendre en compte la situation décrite précédemment, quelles conséquences garderions-nous dans l'ontologie ?

En se fondant sur les contraintes et les approches existantes liées à la révision d'ontologies, les objectifs de cette thèse sont de faire une étude approfondie des possibilités d'extension ou de reformulation des opérateurs de révision existants respectant les postulats AGM pour des ontologies d'expressivité $\mathcal{SHIQ}$ et de proposer de nouveaux algorithmes palliant les inconvénients inhérents à ces opérateurs, mais aussi des solutions pratiques permettant d'implémenter un prototype qui valide les résultats théoriques. Pour ce faire, cette thèse se divise en trois parties, à savoir :

La **partie I** expose le cadre théorique de la thèse. Le *chapitre 1* présentera les notions nécessaires à la compréhension de la révision d'ontologies et à leurs utilisations. Ces notions de base portent sur les logiques de description utilisées pour exprimer des ontologies, ainsi qu'un algorithme de tableau utilisé pour vérifier la cohérence des ontologies. Nous avons également étudié, dans le *chapitre 2*, les approches existantes de la révision d'ontologies pour avoir un aperçu des problèmes restant à solutionner. Ces approches existantes peuvent être divisées en deux catégories à savoir les approches fondées sur les syntaxes et celles fondées sur les modèles. Les approches syntaxiques manipulent directement des entités syntaxiques telles que les formules d'une base de connaissances. Contrairement aux approches syntaxiques, les approches sémantiques manipulent des modèles d'ontologies plutôt que leurs entités syntaxiques.

La **partie II** décrit nos différentes contributions en terme de révision d'ontologies en $\mathcal{SHIQ}$. Nous avons proposé un nouvel algorithme de tableau pour la révision des ontologies exprimées en une logique de description expressive, à savoir $\mathcal{SHIQ}$. En créant ce nouvel algorithme de tableau pour une ontologie en $\mathcal{SHIQ}$, nous avons défini la notion des *modèles de graphe* finis, selon des *modèles d'arbre* dans le chapitre *chapitre 3* ou des *modèles de forêt* dans le *chapitre 4* pour le traitement des ontologies sans individus ou avec individus, respectivement, afin de représenter un ensemble éventuellement infini de modèles d'une ontologie d'expressivité $\mathcal{SHIQ}$. Ces structures finies équipées d'un pré-ordre total permettent de déterminer la différence sémantique entre deux ontologies représentées comme deux ensembles de modèles. En effet, la révision d'une ontologie $\mathcal{O}$ par une autre ontologie $\mathcal{O}'$ peut être réduite à la sélection des modèles "appropriées" à partir de l'ensemble de tous les modèles admis par $\mathcal{O}'$ de telle façon que les modèles sélectionnés

soient les plus proches des modèles de $\mathcal{O}$. Une telle sélection est fondée sur la distance entre des *modèles de graphe* finis qui est destiné à aborder le principe du changement minimal. Enfin, les modèles sélectionnés sont employés afin de construire une ontologie de révision de $\mathcal{O}$ par $\mathcal{O}'$. Notons que nous avons montré que l'ontologie résultante est toujours exprimable en $\mathcal{SHIQ}$ et la taille de celle-ci est triplement exponentielle en fonction des tailles de $\mathcal{O}$ et $\mathcal{O}'$ dans le pire des cas.

Enfin, dans la **partie III**, le *chapitre 5* concerne les techniques d'optimisation applicables aux forêts de complétion que nous avons proposées pour réduire la complexité. Et, le *chapitre 6* traite des premières expérimentations réalisées pour évaluer notre moteur de révision et présente l'outil "*open source*" implémenté accessible en ligne.

Nous finissons ce mémoire par une conclusion générale rappelant tout le travail accompli et par les perspectives envisageables.

Première partie

État de l'art

Chapitre 1

Contexte de l'étude

Dans ce chapitre, nous allons délimiter le contexte général de cette étude en présentant tout d'abord les fondements des logiques de description ; et, plus précisément, la logique expressive $\mathcal{SHIQ}$ pour laquelle nous essayons de solutionner la problématique de la révision d'ontologies. Cette dernière ne pouvant se faire sans la connaissance des techniques de raisonnement, une description générale des algorithmes de tableau sera introduite.

1.1 Logiques de description

Les **logiques de description** (LD) sont une famille de langages de représentation de connaissances qui peuvent être utilisés pour représenter la connaissance terminologique d'un domaine d'application d'une manière formelle et structurée [Baader and Nutt, 2003]. Ces langages ont été introduits dans les années 80, et leur développement fut fortement influencé par les travaux sur la logique des prédicats, les schémas (frames) [Minsky, 1981] et les réseaux sémantiques [Fournier-Viger, 2005].

Pour introduire formellement quelques notions de base des logiques de description, nous commencerons par celles qui contiennent un sous-ensemble des constructeurs suivants : conjonction ($\sqcap$), disjonction ($\sqcup$), restriction universelle ($\forall R.C$), restriction existentielle ($\exists R.C$), restrictions de cardinalité supérieure et inférieure ($\geq n.R$, $\leq n.R$) et négation primitive (ou complète).

C, D	$\rightarrow$	$\top \mid$	(concept universel ou top)
		$\bot \mid$	(concept vide ou bottom)
		$A \mid$	(nom de concept)
		$\neg A \mid$	(négation primitive)
		$C \sqcap D \mid$	(conjonction de concepts)
		$C \sqcup D \mid$	(disjonction de concepts)
		$\forall R.C \mid$	(restriction universelle)
		$\exists R.C \mid$	(restriction existentielle)
		$\geq n.R \mid$	(restriction de cardinalité supérieure)
		$\leq n.R \mid$	(restriction de cardinalité inférieure)
		$\neg C \mid$	(négation complète)

FIGURE 1.1 – Syntaxe des Descriptions de Concept

Les logiques de description utilisent les notions de *concept*, *rôle* et *individu*. Un concept correspond à une "classe d'éléments", les rôles aux "liens entre les éléments" et les individus, quant à eux, aux éléments d'un univers donné. Le nom de logique de description se rapporte à la *description de concept* utilisée pour décrire un domaine. La description de concept est définie récursivement à partir d'un ensemble de *noms de concept* $\mathbf{C}$, d'un ensemble de *noms de rôle* $\mathbf{R}$ et de l'ensemble des constructeurs que ce langage possède. Dans ce mémoire, les descriptions de concept se composent selon les règles de syntaxe décrites dans la figure 1.1. Par exemple, le concept Homme représente la classe de tous les hommes, et le concept Femme la classe de toutes les femmes ; le rôle marié qui relie Homme à Femme peut être représenté aussi par le concept $\mathsf{Homme} \sqcap \exists \mathsf{marié}.\mathsf{Femme}$, ceci représente la classe des hommes qui sont mariés avec des femmes. Par convention, nous désignons, s'il n'y a pas d'indication particulière, les *concepts atomiques* (*noms de concept*) dans $\mathbf{C}$ par les lettres A, B, les *rôles atomiques* (*noms de rôle*) dans $\mathbf{R}$ par les lettres R, S, les *concepts complexes* par les lettres C, D. Pour un concept atomique A, le concept $A \sqcap \neg A$ est équivalent au concept $\bot$.

Pour faciliter la construction, nous supposons que tout concept est sous la forme normale négative (*negation normal form* ou NNF), *i.e.* la négation se trouve seulement devant des noms de concept. Tout concept peut être transformé en un concept équivalent dans NNF

en utilisant les lois de De Morgan et de certaines équivalences [Horrocks *et al.*, 1999]. Pour un concept C, nous utilisons $\mathsf{nnf}(C)$ et $\dot{\neg}C$ pour désigner respectivement la NNF de C et de $\neg C$. Notons que la fonction $\mathsf{nnf}(C)$ peut être calculée en temps polynomial par rapport à la taille de C [Baader *et al.*, 2007].

La sémantique des logiques de description est spécifiée par des interprétations. Une interprétation $\mathcal{I}$ est une paire $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}} \rangle$ où $\Delta^{\mathcal{I}}$ est un ensemble non vide appelé *domaine* et $.^{\mathcal{I}}$ est une fonction d'interprétation qui associe chaque concept atomique A à un sous-ensemble $A^{\mathcal{I}}$ de $\Delta^{\mathcal{I}}$, chaque rôle atomique P à une relation binaire $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, et chaque nom d'individu a à un élément $a^{\mathcal{I}}$ de $\Delta^{\mathcal{I}}$.

1.1.1 Base de connaissances

Dans la plupart des logiques de description, une *base de connaissances* (BC) est divisée en deux parties. La première, concernant l'aspect terminologique dit **TBox**, décrit les connaissances générales d'un domaine alors que la seconde, concernant l'aspect assertionnel dit **ABox**, représente un monde possible qui est censé obéir à la **TBox**. Une TBox comprend la définition des concepts et des rôles alors qu'une ABox décrit la relation d'appartenance des individus à des concepts et rôles. Plusieurs ABox peuvent être associées à une même TBox. Chacune d'elles représente un monde constitué d'individus et utilise les concepts et rôles de la TBox pour le décrire. [Fournier-Viger, 2005].

Formellement, une TBox $\mathcal{T}$ est un ensemble fini, éventuellement vide, d'*axiomes terminologiques* de la forme $C \sqsubseteq D$ ou $C \equiv D$ où C et D sont des concepts atomiques ou complexes. Un axiome $C \sqsubseteq D$ (C est subsumé par D) qui est un GCI (*"General Concept Inclusion Axiom"* en anglais) permet d'exprimer des relations d'inclusion, alors que $C \equiv D$ (C est équivalent à D) est une notation pour $C \sqsubseteq D$ et $D \sqsubseteq C$. Par exemple, $Homme \sqsubseteq Humain$ peut être utilisé pour stipuler que "Tous les hommes doivent être humains". Et $Humain \equiv Homme \sqcup Femme$ définit le concept $Humain$ qui est "l'ensemble des hommes et des femmes".

Une ABox $\mathcal{A}$ est un ensemble fini, éventuellement vide, d'*assertions* sur les individus : des *assertions de concept* de la forme $C(a)$ et des *assertions de rôle* de la forme $R(a, b)$ où

C est un concept général, R est un rôle, et a, b sont des noms d'individu. Par exemple, $Homme(Tom)$ dit que "Tom est un homme", et $marié(Tom, Sophie)$ stipule que "Tom est marié avec Sophie".

Formellement, une base de connaissances est une paire $(\mathcal{T}, \mathcal{A})$ d'une TBox $\mathcal{T}$ et d'une ABox $\mathcal{A}$. En représentant une ontologie comme une base de connaissances de logiques de description constituée d'une TBox et d'une ABox, les logiques de description fournissent aux applications ontologiques des fondements logiques et des mécanismes de raisonnement. Dans le reste de ce mémoire, nous ne ferons pas la distinction entre une ontologie et une base de connaissances.

1.1.2 Différentes logiques de description

Les logiques de description ont une base commune enrichie de différentes extensions. Le tableau 1.1 montre les constructeurs correspondants aux différentes logiques de description ayant des concepts complexes composés de concepts atomiques, de même pour les rôles. La première colonne contient la lettre qui désigne le constructeur, la deuxième sa syntaxe d'utilisation et la dernière sa sémantique. Nous utilisons $|S|$ pour désigner la cardinalité d'un ensemble S.

Le langage $\mathcal{FL}^-$ [Brachrnan and Levesque, 1984], qui représente l'une des premières logiques de description, permet l'utilisation des restrictions universelles, de la conjonction de concepts, et des restrictions existentielles de la forme $\exists R.\top$. La logique $\mathcal{AL}$ [Schmidt-Schauß and Smolka, 1991] a étendu la logique $\mathcal{FL}^-$ en y ajoutant la négation des concepts atomiques. Cette logique peut être considérée comme la logique de base des autres logiques de description. Les logiques de description qui existent sont des combinaisons des différents éléments du tableau 1.1. Par exemple, si l'on rajoute la négation complète $\mathcal{C}$ à la logique $\mathcal{AL}$, on obtient la logique $\mathcal{ALC}$. Remarquons que, parmi les logiques existantes, il y a des logiques qui sont équivalentes, *e.g.* $\mathcal{ALC}$ [Schmidt-Schauß and Smolka, 1991] et $\mathcal{ALUE}$.

Lettre	Constructeur	Syntaxe	Sémantique
$\mathcal{AL}$	concept universel	$\top$	$\Delta^{\mathcal{I}}$
$\mathcal{AL}$	concept vide	$\bot$	$\emptyset$
$\mathcal{C}$	négation de concepts (non nécessairement primitifs)	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$\mathcal{AL}$	conjonction de concepts	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$\mathcal{U}$	disjonction de concepts	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\mathcal{AL}$	restriction universelle	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x,y) \in R^{\mathcal{I}}$ *implique* $y \in C^{\mathcal{I}}\}$
$\mathcal{E}$	restriction existentielle typée	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x,y) \in R^{\mathcal{I}}$ *et* $y \in C^{\mathcal{I}}\}$
$\mathcal{N}$	restriction de cardinalité	$\geq nR$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert\{x \mid (x,y) \in R^{\mathcal{I}}\}\rvert \geq n\}$
		$\leq nR$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert\{x \mid (x,y) \in R^{\mathcal{I}}\}\rvert \leq n\}$
$\mathcal{Q}$	restriction de cardinalité qualifiée	$\geq nR.C$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert\{x \mid (x,y) \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\}\rvert \geq n\}$
		$\leq nR.C$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert\{x \mid (x,y) \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\}\rvert \leq n\}$
$\mathcal{O}$	un-de	$\{a_1, \ldots, a_n\}$	$\{x \in \Delta^{\mathcal{I}} \mid x = a_i^{\mathcal{I}}$ *pour un* $a_i\}$
$\mathcal{B}$	rôle filler	$\exists R.\{a\}$	$\{x \in \Delta^{\mathcal{I}} \mid (x, a_i^{\mathcal{I}}) \in R^{\mathcal{I}}\}$
$\mathcal{AL}$	nom de rôle	R	$R^{\mathcal{I}}$
$\mathcal{R}$	conjonction de rôles	$R \sqcap S$	$R^{\mathcal{I}} \cap S^{\mathcal{I}}$
$\mathcal{I}$	rôles inverses	R^-	$\{(x,y) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (y,x) \in R^{\mathcal{I}}\}$
$\mathcal{H}$	hiérarchie de rôles	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
$\mathcal{R}^+$	transitivité de rôles	R^+	Plus petite relation transitive contenant $R^{\mathcal{I}}$

TABLEAU 1.1 – Langages des LD

1.1.3 Inférences

Afin d'introduire la notion d'inférence s'effectuant au niveau terminologique ou assertionnel, nous utilisons $\mathcal{I} \models \varphi$ pour indiquer qu'une interprétation $\mathcal{I}$ satisfait un axiome ou une assertion φ. Par conséquent, $\mathcal{I}$ satisfait un axiome $C \sqsubseteq D$ (notation $\mathcal{I} \models C \sqsubseteq D$) si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, $\mathcal{I}$ satisfait une assertion de concept $C(a)$ (notation $\mathcal{I} \models C(a)$) si $a^{\mathcal{I}} \in C^{\mathcal{I}}$, et $\mathcal{I}$ satisfait une assertion de rôle $R(a,b)$ (notation $\mathcal{I} \models R(a,b)$) si $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. On dit que $\mathcal{I}$ est un modèle d'une TBox $\mathcal{T}$ (désignée par $\mathcal{I} \models \mathcal{T}$) si et seulement si $\mathcal{I}$ satisfait chaque axiome de $\mathcal{T}$, et $\mathcal{I}$ est un modèle d'une ABox $\mathcal{A}$ (désignée par $\mathcal{I} \models \mathcal{A}$) si et seulement si $\mathcal{I}$ satisfait chaque assertion de $\mathcal{A}$. Une interprétation $\mathcal{I}$ est un modèle d'une base de connaissances $(\mathcal{T}, \mathcal{A})$ si $\mathcal{I}$ est un modèle de $\mathcal{T}$ et de $\mathcal{A}$, *i.e.*, $\mathcal{I} \models \mathcal{T}$ et $\mathcal{I} \models \mathcal{A}$. Une base de connaissances est *cohérente* si et seulement si elle a au moins un modèle.

Les tâches de raisonnement, qui font référence à la capacité de réaliser des inférences depuis une base de connaissances $K = (\mathcal{T}, \mathcal{A})$, incluent les principaux cas d'inférence suivants

[Baader and Nutt, 2003] :

- **Satisfiabilité** : un concept C est satisfiable par rapport à la TBox $\mathcal{T}$ s'il existe un modèle $\mathcal{I}$ de K tel que $C^{\mathcal{I}} \neq \emptyset$.
- **Subsomption** : un concept C est subsumé par un concept D par rapport à la TBox $\mathcal{T}$ si et seulement si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ pour tous les modèles $\mathcal{I}$ de K.
- **Vérification d'instance** : un individu a de l'ABox $\mathcal{A}$ est une *instance* d'un concept C par rapport à la TBox $\mathcal{T}$ si et seulement si $a^{\mathcal{I}} \in C^{\mathcal{I}}$ pour tous les modèles $\mathcal{I}$ de K.
- **Vérification de relation** : un rôle R est une relation d'un individu a à un autre b si et seulement si $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ pour tous les modèles $\mathcal{I}$ de K.
- **Cohérence** : l'ABox $\mathcal{A}$ est cohérente par rapport à la TBox $\mathcal{T}$ si et seulement s'il existe un modèle $\mathcal{I}$ de $\mathcal{T}$ et de $\mathcal{A}$.

Illustrons ces différents cas par l'exemple 1 lors des tâches de raisonnement.

Exemple 1. Soit K une base de connaissances $(\mathcal{T}, \mathcal{A})$ où

$\mathcal{T} = \{\mathsf{Joueur} \sqsubseteq \mathsf{Personne} \sqcap \forall \mathsf{joue}.\mathsf{Match}, \mathsf{Arbitre} \sqsubseteq \neg \mathsf{Joueur}\}$,
$\mathcal{A} = \{\mathsf{Joueur}(\mathsf{Paul}), \mathsf{Personne}(\mathsf{Peter}), \mathsf{joue}(\mathsf{Paul}, \mathsf{m1})\}$.

Deux axiomes de la TBox $\mathcal{T}$ énoncent qu'un joueur est une personne qui joue des matchs et qu'un arbitre n'est pas un joueur. Les assertions de l'ABox signifient que *Paul* est un joueur qui joue le match $m1$, et *Peter* est une personne. Soit $\mathcal{I}$ une interprétation telle que $\Delta^{\mathcal{I}} = \{\mathsf{Paul}, \mathsf{Peter}, \mathsf{m1}\}$, et $\mathsf{Personne}^{\mathcal{I}} = \{\mathsf{Paul}, \mathsf{Peter}\}$, $\mathsf{Joueur}^{\mathcal{I}} = \{\mathsf{Paul}\}$, $\mathsf{Match}^{\mathcal{I}} = \{\mathsf{m1}\}$, $\mathsf{Arbitre}^{\mathcal{I}} = \emptyset$, $\mathsf{joue}^{\mathcal{I}} = \{(\mathsf{Paul}, \mathsf{m1})\}$. Par inférences, on peut vérifier que :

— $\mathcal{I}$ satisfait chaque axiome de la TBox $\mathcal{T}$ et chaque assertion de l'ABox $\mathcal{A}$. Alors, $\mathcal{I}$ est un modèle de $\mathcal{T}$ et de $\mathcal{A}$, et K est donc cohérente ;

— tous les concepts $\mathsf{Joueur}, \mathsf{Personne}, \mathsf{Arbitre}$, et Match sont satisfiables par rapport à la TBox $\mathcal{T}$;

— le concept Joueur est subsumé par le concept $\mathsf{Personne}$ tandis que le concept $\mathsf{Arbitre}$ n'est pas subsumé par le concept Joueur pour le modèle $\mathcal{I}$ de K ;

— l'individu Paul est une instance du concept Personne mais l'individu Peter n'est pas une instance du concept Joueur pour le modèle $\mathcal{I}$ de K ;

— le rôle joue est une relation entre les individus Paul et m1 tandis qu'une assertion joue(Peter, m1) n'est pas vraie pour le modèle $\mathcal{I}$ de K.

Notons qu'il y a une relation étroite entre les inférences de satisfiabilité et de subsomption. En effet, pour les langages LD qui comportent le constructeur de négation complète (*e.g.*, $\mathcal{ALC}$), la subsomption peut être réduite à la satisfiabilité car $C \sqsubseteq D$ est équivalent à $C \sqcap \neg D \equiv \bot$. Inversement, le problème de satisfiabilité peut être réduit au problème de subsomption car une description de concept C est insatisfiable si et seulement si $C \sqsubseteq \bot$.

1.2 Ontologie d'expressivité $\mathcal{SHIQ}$

La logique $\mathcal{SHIQ}$ est fondée sur une extension de la bien connue LD $\mathcal{ALC}$ en incluant la fermeture transitive des rôles primitifs [Sattler, 1996]. À partir du tableau 1.1, la logique $\mathcal{ALC}$ augmentée par $\mathcal{R}^+$ est notée $\mathcal{S}$ [Horrocks *et al.*, 1999]. Cette LD de base est ensuite étendue avec la hiérarchie de rôles ($\mathcal{H}$), les rôles inverses ($\mathcal{I}$) et les restrictions de cardinalité qualifiées ($\mathcal{Q}$).

Soient $\mathbf{R}$ un ensemble non vide de *noms de rôle* et $\mathbf{R}_+ \subseteq \mathbf{R}$ un ensemble de noms de rôle transitif, nous utilisons $\mathbf{R}_\mathsf{I} = \{R^- \mid R \in \mathbf{R}\}$ pour désigner un ensemble de rôles inverses. Chaque élément de $\mathbf{R} \cup \mathbf{R}_\mathsf{I}$ s'appelle un $\mathcal{SHIQ}$-rôle. Pour simplifier les notations concernant les rôles inverses imbriqués, nous définissons une fonction $\mathsf{Inv}(S) = R^-$ si $S = R$; et $\mathsf{Inv}(S) = R$ si $S = R^-$ où $R \in \mathbf{R}$. Une *inclusion d'axiome de rôles* est de la forme $R \sqsubseteq S$ pour deux (éventuellement inverse) $\mathcal{SHIQ}$-rôles R et S. Une *hiérarchie de rôles* $\mathcal{R}$ est un ensemble fini d'inclusion d'axiomes de rôles. Une relation de sous-rôles $\sqsubseteq\!\!\!\!\sqsubseteq$ est définie comme la fermeture transitive-réflexive de $\sqsubseteq$ sur $\mathcal{R}^+ = \mathcal{R} \cup \{\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \mid R \sqsubseteq S \in \mathcal{R}\}$. Nous définissons une fonction $\mathsf{Trans}(R)$ qui retourne true si et seulement si (ssi) R est un rôle transitif. Plus précisément, $\mathsf{Trans}(R) = \mathit{true}$ ssi $R \in \mathbf{R}_+$ ou $\mathsf{Inv}(R) \in \mathbf{R}_+$. Un rôle R est *simple* par rapport à $\mathcal{R}$ si $\mathsf{Trans}(R) = \mathsf{false}$. Une interprétation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ se compose d'un ensemble non vide $\Delta^\mathcal{I}$ (*domaine*) et une fonction $\cdot^\mathcal{I}$ qui associe un nom de rôle à

un sous-ensemble de $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ telle que $R^{-\mathcal{I}} = \{\langle x, y\rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \langle y, x\rangle \in R^{\mathcal{I}}\}$ pour tout $R \in \mathbf{R}$, et $\langle x, z\rangle \in S^{\mathcal{I}}, \langle z, y\rangle \in S^{\mathcal{I}}$ implique $\langle x, y\rangle \in S^{\mathcal{I}}$ pour chaque $S \in \mathbf{R}_+$. Une interprétation $\mathcal{I}$ est un modèle de $\mathcal{R}$, écrit $\mathcal{I} \models \mathcal{R}$ si et seulement si $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ pour chaque $R \sqsubseteq S \in \mathcal{R}$.

Soit $\mathbf{C}$ un ensemble non vide de *noms de concept*. L'ensemble de $\mathcal{SHIQ}$-concepts est défini inductivement comme le plus petit ensemble contenant tout C dans $\mathbf{C}$, $\top$, $C \sqcap D$, $C \sqcup D$, $\neg C$, $\exists R.C$, $\forall R.C$, $(\leq n\, S.C)$ et $(\geq n\, S.C)$ où n est un entier positif, C et D sont $\mathcal{SHIQ}$-concepts, R est un $\mathcal{SHIQ}$-rôle et S est un rôle simple par rapport à une hiérarchie de rôles. Nous écrivons $\bot$ pour $\neg\top$. La fonction d'interprétation $\cdot^{\mathcal{I}}$ d'une interprétation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ associe chaque nom de concept à un ensemble de $\Delta^{\mathcal{I}}$ comme des associations dans le tableau 1.1.

Un axiome $C \sqsubseteq D$ s'appelle une inclusion générale de concepts (GCI) où C, D sont des $\mathcal{SHIQ}$-concepts (éventuellement complexes), et un ensemble fini de GCIs s'appelle une TBox $\mathcal{T}$. Une interprétation $\mathcal{I}$ satisfait une GCI $C \sqsubseteq D$, écrit $\mathcal{I} \models (C \sqsubseteq D)$, si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Une interprétation $\mathcal{I}$ est un modèle de $\mathcal{T}$, noté $\mathcal{I} \models \mathcal{T}$, si et seulement si $\mathcal{I}$ satisfait chaque GCI dans $\mathcal{T}$.

Soit $\mathbf{I}$ un ensemble de noms d'individu. Une assertion est de la forme $C(a)$ (assertion de concept), $R(a, b)$ (assertion de rôle), ou $a \neq b$ pour $a, b \in \mathbf{I}$, un $\mathcal{SHIQ}$-rôle R et un $\mathcal{SHIQ}$-concept C. Une ABox se compose d'un ensemble fini d'assertions. Pour une interprétation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, un élément $x \in \Delta^{\mathcal{I}}$ s'appelle une instance d'un concept C si et seulement si $x \in C^{\mathcal{I}}$. Pour les ABox, la fonction d'interprétation $\cdot^{\mathcal{I}}$ de $\mathcal{I}$ associe chaque $a \in \mathbf{I}$ à certains éléments $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. Une interprétation $\mathcal{I}$ satisfait une assertion $C(a)$ (resp. $R(a, b)$, et $a \neq b$) si $a^{\mathcal{I}} \in C^{\mathcal{I}}$ (resp. $\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$, et $a^{\mathcal{I}} \neq b^{\mathcal{I}}$). $\mathcal{I}$ satisfait une ABox $\mathcal{A}$ si et seulement si elle satisfait chaque assertion dans $\mathcal{A}$. Une telle interprétation s'appelle un *modèle* de $\mathcal{A}$, désigné par $\mathcal{I} \models \mathcal{A}$.

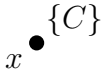
Nous utilisons $\mathcal{O} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ pour désigner une *ontologie* en $\mathcal{SHIQ}$, où $\mathcal{T}$ est une TBox en $\mathcal{SHIQ}$, $\mathcal{R}$ est une hiérarchie de rôles en $\mathcal{SHIQ}$, et $\mathcal{A}$ est une ABox en $\mathcal{SHIQ}$. Une ontologie $\mathcal{O} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ est cohérente s'il y a un modèle $\mathcal{I}$ de $\mathcal{T}$, $\mathcal{R}$ et $\mathcal{A}$, *i.e.*, $\mathcal{I} \models \mathcal{T}$, $\mathcal{I} \models \mathcal{R}$ et $\mathcal{I} \models \mathcal{A}$. De plus, nous utilisons $\mathsf{Mod}(\mathcal{O})$ pour désigner tous les modèles, et

$\mathcal{S}(\mathcal{O}) = \mathbf{R} \cup \mathbf{C} \cup \mathbf{I}$ pour désigner la signature d'une ontologie $\mathcal{O}$ où $\mathbf{C}$ est l'ensemble des noms de concept se trouvant dans $\mathcal{O}$, $\mathbf{R}$ est l'ensemble des noms de rôle se trouvant dans $\mathcal{O}$, et $\mathbf{I}$ est l'ensemble des noms d'individu se trouvant dans $\mathcal{A}$.

1.3 Introduction aux algorithmes de tableau pour LD

Afin de vérifier la satisfiabilité d'un concept, dans plusieurs LD expressives, il existe des implémentations efficaces à base de tableaux (algorithmes de tableau) [Horrocks, 1998; Haarslev and Möller, 2001]. Brièvement, un algorithme de tableau qui est une *procédure de décision* tente de construire un *graphe* dont les nœuds représentent les individus d'une interprétation. Chaque nœud x est étiqueté avec un ensemble de concepts $L(x)$, à savoir celui dont il est supposé être une instance. Chaque arête $\langle x, y\rangle$ entre deux nœuds x, y est étiquetée avec un ensemble de noms de rôle $L(\langle x, y\rangle)$, à savoir celui qui représente des relations entre deux individus correspondants à x et y. Lorsque $\langle x, y\rangle$ est étiquetée avec un ensemble contenant le nom de rôle P, x est dit P-prédécesseur (P-voisin) de y, y est dit P-successeur (P-voisin) de x.

Intuitivement, pour vérifier la satisfiabilité d'un concept C par rapport à une TBox $\mathcal{T}$, un algorithme de tableau commence par une instance x_0 de C. C'est-à-dire qu'un graphe constitué d'un nœud racine x avec C comme étiquette de nœud (écrit $L(x) = \{C\}$) :



Puis, l'algorithme décompose syntaxiquement les concepts dans les étiquettes de nœud pour déduire de nouvelles contraintes sur le modèle de C à construire, et éventuellement génère de nouveaux individus, *i.e.* de nouveaux nœuds. Pour cela, l'algorithme applique des règles d'expansion correspondant à chaque LD [Baader and Sattler, 2001]. Les règles d'expansion correspondent aux constructeurs présentés dans le tableau 1.2. La première colonne montre des nœuds avec leurs étiquettes tandis que la deuxième est le résultat de l'application d'une règle (*cf.* deuxième colonne) sur ces nœuds.

Tout d'abord, il existe une règle qui ajoute le concept $(\neg C_i \sqcup D_i)$ dans $L(x)$ pour chaque

$\{C_1 \sqcap C_2, \ldots\}$ x	$\sqcap$-règle	$\{C_1 \sqcap C_2, C_1, C_2, \ldots\}$ x
$\{C_1 \sqcup C_2, \ldots\}$ x	$\sqcup$-règle	$\{C_1 \sqcap C_2, C, \ldots\}$ x pour $C \in \{C_1, C2\}$
$\{\exists R.C, \ldots\}$ x	$\exists$-règle	$\{\exists R.C, \ldots\}$ x R y $\{C, \ldots\}$
$\{\forall R.C, \ldots\}$ x R y $\{\ldots\}$	$\forall$-règle	$\{\forall R.C, \ldots\}$ x R y $\{C, \ldots\}$
$\{(\geq nR), \ldots\}$ x x n'a pas plus de n R-successeurs	$\geq$-règle	$\{(\geq nR), \ldots\}$ x R y_1 $\{\}$ y_2 $\{\}$ ..... y_n $\{\}$
$\{(\leq nR), \ldots\}$ x R $\{\}$ $\{\}$ ..... $\{\}$ $> n$	$\leq$-règle	$\{(\leq nR), \ldots\}$ x R $\{\}$ $\{\}$ $\{\}$ $\{\}$ fusionner 2 R-successeurs

TABLEAU 1.2 – Illustration des règles d'expansion pour des algorithmes de tableau

GCI $C_i \sqsubseteq D_i \in \mathcal{T}$ et pour chaque nœud x. Ensuite, si $(D \sqcap E) \in L(x)$ d'un nœud x du graphe, l'algorithme doit ajouter les concepts D et E dans $L(x)$. Pour le cas où $\exists R.D \in L(x)$, il génère un nouveau nœud y qui est un R-successeur de x, et fixe $L(y) = \{D\}$. Si un nœud x a un R-successeur y et $\forall R.F \in L(x)$, alors F est ajouté dans $L(y)$. Un algorithme de tableau peut traiter des cas non-déterministes comme $(D \sqcup E) \in L(x)$. Il doit alors choisir d'ajouter soit D, soit E, dans $L(x)$. Si un nœud x n'a pas plus de n R-successeurs et $(\geq nR) \in L(x)$, l'algorithme génère n nouveaux R-successeurs de x. En revanche, si un nœud x a plus de n R-successeurs et $(\leq nR) \in L(x)$, l'algorithme fusionne deux R-successeurs de x jusqu'à ce que x ait seulement n R-successeurs. Le processus de

l'algorithme de tableau peut être réalisé de manière exhaustive sans rencontrer de nœud qui soit à la fois un concept et sa négation dans son étiquette (ce cas de figure est appelé *clash*) ; l'algorithme conclut alors que le concept d'entrée est satisfiable, et sinon il est insatisfiable.

Pour garantir la **terminaison** de la construction de graphe, l'algorithme de tableau utilise une technique appelée *technique de blocage* (*blocking technique* en anglais). L'idée sous-jacente du mécanisme de blocage est de détecter des "boucles" qui sont des éléments répétés d'un graphe. Quand un nœud bloqué est détecté sur chaque branche, l'algorithme s'arrête sur cette branche et continue sur les autres jusqu'à ce qu'aucune règle est applicable sur ce graphe. De telles boucles représentent des parties infinies d'un modèle. Par exemple, vérifions un concept $C \sqcap \exists S.C$ par rapport à une TBox :

$$\mathcal{T} = \{\top \sqsubseteq \forall R.(\exists S.C), \top \sqsubseteq \exists R.C\}.$$

En appliquant l'algorithme de tableau, un graphe est initialisé par une racine x avec son étiquette $L(x)$ contenant le concept $C \sqcap \exists S.C$. Deux concepts $\bot \sqcup \forall R.(\exists S.C)$ et $\bot \sqcup \exists R.C$ provenant de la TBox $\mathcal{T}$ sont ajoutés dans $L(x)$. Puis les $\sqcap$-, $\sqcup$-règles sont appliquées sur ces concepts pour obtenir une étiquette de x comme dans le Figure 1.2. Notons que la $\sqcup$-règle ne peut choisir le concept $\bot$ à ajouter dans l'étiquette afin d'éviter un clash. Ensuite, l'algorithme applique les $\exists$-,$\forall$-règles sur les concepts ajoutés dans $L(x)$ pour générer deux nouveaux nœuds y et z. Les étiquettes $L(y)$ et $L(z)$ sont remplies de la même manière que $L(x)$. L'algorithme n'a pas besoin d'appliquer les $\exists$-,$\forall$-règles sur les concepts dans $L(y)$ et $L(z)$ car cela crée des éléments répétés du graphe sachant que $L(y) \subset L(x)$ et $L(z) \subset L(x)$. Dans ce cas, x est le nœud bloquant et y, z sont les nœuds bloqués. Alors, le résultat de la figure 1.2 est un graphe sans clash construit par l'algorithme de tableau et le concept $C \sqcap \exists S.C$ est donc satisfiable par rapport à la TBox $\mathcal{T}$.

La **correction** de l'algorithme de tableau est assurée telle façon que si cet algorithme peut construire un graphe sans avoir généré de clash, alors le concept d'entrée est satisfiable, *i.e.* un modèle du concept d'entrée (et de la TBox) peut être construit. Un tel modèle peut être construit en "dénouage" ("unraveling" anglais) du graphe construit. Ce processus génère des descendants de nœuds bloqués en répliquant les descendants de nœuds de blocage. Le

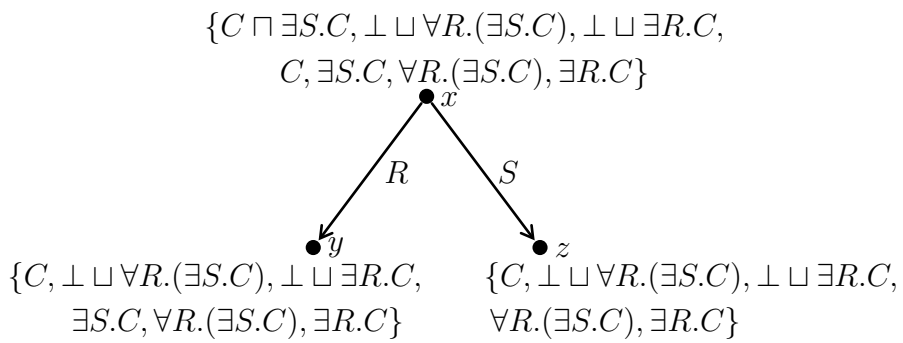


FIGURE 1.2 – Exemple d'algorithme de tableau

modèle obtenu est un graphe qui peut être une structure infinie.

Inversement, si le concept d'entrée est satisfiable, alors l'algorithme peut construire un graphe sans avoir généré de clash. Dans ce cas, la **complétude** peut être assurée en utilisant un modèle de concepts d'entrée (et de la TBox) pour diriger l'application des règles non-déterministes et choisir des concepts corrects (ou successeurs corrects) afin de construire un graphe qui ne contient aucun clash.

Lors de l'application de l'algorithme de tableau sur des ontologies exprimables dans la logique expressive $\mathcal{SHIQ}$, Horrocks *et al.* [Horrocks *et al.*, 1999] ont proposé un ensemble de règles et de techniques pour pouvoir traiter la transitivité de rôles, la hiérarchie de rôles, les rôles inverses et les restrictions de cardinalité qualifiées. Dans le pire des cas, la complexité du raisonnement de l'algorithme de tableau pour $\mathcal{SHIQ}$ est EXPTIME-complet [Tobies, 2001]. Cependant, en pratique, le comportement des algorithmes de tableau est souvent acceptable [Baader *et al.*, 2003].

1.4 Conclusion

Nous avons présenté les fondements des logiques de description ainsi que la logique spécifique $\mathcal{SHIQ}$ qui est une logique de description expressive. Nous avons aussi introduit la base des algorithmes de tableau qui seront utilisés dans la partie II pour solutionner la problématique de la révision d'ontologies exprimées en $\mathcal{SHIQ}$. Dans le chapitre suivant, nous allons présenter des approches existantes pour la révision d'ontologies exprimées par

des logiques de description.

Chapitre 2

Révision d'ontologies

Le problème de la révision d'ontologies exprimables en OWL-DL (appelée ontologies en LD) est étroitement lié au problème de la révision de croyances qui a été largement discuté dans la littérature. L'approche la plus connue dans la révision de croyances est le paradigme AGM de Alchourrón, Gärdenfors et Makinson [Alchourrón *et al.*, 1985]. Dans ce paradigme AGM, des *croyances* sont caractérisées par des phrases d'un langage formel L, et des *ensembles de croyances* par des ensembles déductivement fermés de phrases. Le processus de la révision de croyances est modélisé comme une fonction $\star$ qui associe un ensemble de croyances K et une phrase A à un nouvel ensemble de croyances $K \star A$. Intuitivement, $K \star A$ correspond à K modifié au minimum avec A intégré et qui reste cohérente. En particulier, si A est logiquement cohérente avec K, A est simplement ajoutée à K. De plus, on désigne par $Cn(K)$ l'ensemble de toutes les conséquences de K telles que $K \subseteq Cn(K)$ (inclusion), $Cn(K) = Cn(Cn(K))$ (itération) et $Cn(K) \subseteq Cn(K')$ si $K \subseteq K'$ (monotonicité). Alchourrón, Gärdenfors et Makinson [Alchourrón *et al.*, 1985] ont introduit un ensemble de postulats (appelé postulats d'AGM) que tout processus de révision de croyances devrait respecter, comme suit :

(G$\star$1) $K \star A$ est un ensemble de croyances de L,

(G$\star$2) $A \in K \star A$,

(G$\star$3) $K \star A \subseteq Cn(K \cup \{A\})$,

(G$\star$4) Si $\neg A \notin K$ alors $Cn(K \cup \{A\}) \subseteq K \star A$,

(G⋆5) si A est cohérente alors $K \star A$ est cohérent,

(G⋆6) si $Cn(A) = Cn(B)$ alors $K \star A = K \star B$,

(G⋆7) $K \star (A \wedge B) \subseteq Cn(K \star A \cup \{B\})$,

(G⋆8) si $\neg B \notin K \star \mu$ alors $Cn(K \star A \cup \{B\}) \subseteq K \star (A \wedge B)$.

Les postulats d'AGM sont formulés dans un contexte très général dans lequel il y a très peu d'hypothèses sur L et l'opération de conséquence Cn. Katsuno et Mendelzon [Katsuno and Mendelzon, 1991] ont donc adapté ces postulats dans le cadre de la logique propositionnelle. Soit L un langage formel de logique propositionnelle défini par un ensemble fini de symboles $P := \{p_1, p_2, \ldots, p_n\}$ utilisant les opérateurs usuels $\neg$ (non), $\vee$ (ou), $\wedge$ (et). Une interprétation $\mathcal{I}$ de P est désignée par la forme $(\mathcal{I}(p_1), \ldots, \mathcal{I}(p_n))$ où $\mathcal{I}(p_i) = 1$ $(vrai)$ où $\mathcal{I}(p_i) = 0$ $(faux)$ pour $i = 1, \ldots, n$. Une interprétation $\mathcal{I}$ est un modèle d'une formule $\phi \in L$ si et seulement si ϕ est vraie sous $\mathcal{I}$ par la méthode classique d'une table de vérité. L'ensemble des modèles d'une formule ϕ est noté par $\mathsf{Mod}(\phi)$. Une formule ϕ est satisfiable si $\mathsf{Mod}(\phi) \neq \emptyset$. Deux formules ϕ et ω sont équivalentes (notation $\phi \equiv \omega$) si et seulement si $\mathsf{Mod}(\phi) = \mathsf{Mod}(\omega)$. Une base de connaissances K, qui peut être définie comme un ensemble de formules, valide une formule ϕ (notation $K \models \phi$) si et seulement si chaque modèle de K est aussi un modèle de ϕ, *i.e.* $\mathsf{Mod}(K) \subseteq \mathsf{Mod}(\phi)$.

Les postulats d'AGM sont reformulés par Katsuno et Mendelzon [Katsuno and Mendelzon, 1991] comme suit, où $\omega \circ \mu$ désigne le résultat de la révision d'une formule ω par une nouvelle formule μ.

(R∘1) $\omega \circ \mu$ implique μ,

(R∘2) si $\omega \wedge \mu$ est satisfiable alors $\omega \circ \mu \equiv \omega \wedge \mu$,

(R∘3) si μ est satisfiable alors $\omega \circ \mu$ est aussi satisfiable,

(R∘4) si $\omega_1 \equiv \omega_2$ et $\mu_1 \equiv \mu_2$, alors $\omega_1 \circ \mu_1 = \omega_2 \circ \mu_2$,

(R∘5) $(\omega \circ \mu_1) \wedge \mu_2$ implique $\omega \circ (\mu_1 \wedge \mu_2)$;

(R∘6) Si $(\omega \circ \mu_1) \wedge \mu_2$ est satisfiable, alors $\omega \circ (\mu_1 \wedge \mu_2)$ implique $(\omega \circ \mu_1) \wedge \mu_2$.

Katsuno et Mendelzon [Katsuno and Mendelzon, 1991] ont montré que l'opérateur $\star$ satisfait six postulats **(G⋆1)**-**(G⋆6)** si et seulement si l'opérateur $\circ$ satisfait les quatre postulats

(R∘1)-**(R∘4)** ; de même, les postulats **(G⋆7)** et **(G⋆8)** sont équivalents à **(R∘5)** et **(R∘6)** respectivement. Effectivement, le postulat **(R∘1)** garantit que la nouvelle formule peut être déduit à partir du résultat de révision. Ce postulat correspond aux **(G⋆1)** et **(G⋆2)**. **(R∘2)** dit que la formule initiale n'est pas modifiée dans le résultat de révision si la nouvelle formule est cohérente avec la formule initiale. Cela est similaire à deux postulats **(G⋆3)** et **(G⋆4)** qui, ensemble, expriment aussi la notion de changement minimal dans le cas particulier où il n'y a aucun conflit. **(R∘3)** ou **(G⋆5)** est une condition assurant la cohérence du résultat de révision. **(R∘4)** dit que l'opération de révision doit être indépendante de la syntaxe des formules, soit le même dire que **(G⋆6)**. Quant à **(R∘5)** et **(R∘6)**, ils garantissent le principe de changement minimal, comme **(G⋆7)** et **(G⋆8)** énonçant que pour deux phrases A et B, si le résultat de la révision d'un ensemble de croyances K par la phrase A est $K \circ A$, cohérente avec la phrase B, alors il suffit d'étendre $K \circ A$ avec B pour obtenir $K \circ (A \wedge B)$; c'est-à-dire $K \circ (A \wedge B) = Cn(K \circ A \cup \{B\})$. Pour illustrer **(R∘5)** et **(R∘6)**, supposons qu'il existe une métrique pour mesurer la "distance" entre $\mathsf{Mod}(\omega)$ et une interprétation $\mathcal{I}$. Dans ce cas, le principe de changement minimal est garanti si les modèles de $\omega \circ \mu$ sont ceux de μ qui sont les plus proches de $\mathsf{Mod}(\omega)$ par rapport à la métrique de distance. Le postulat **(R∘5)** dit que si une interprétation $\mathcal{I}$ la plus proche de $\mathsf{Mod}(\omega)$ dans un ensemble $\mathsf{Mod}(\mu_1)$ est sélectionnée et $\mathcal{I}$ appartient aussi à un plus petit ensemble $\mathsf{Mod}(\mu_1 \wedge \mu_2)$ alors $\mathcal{I}$ est la plus proche de $\mathsf{Mod}(\omega)$ dans ce plus petit ensemble $\mathsf{Mod}(\mu_1 \wedge \mu_2)$. Pour le postulat **(R∘6)**, soit une interprétation $\mathcal{I}$, modèle de $\omega \circ (\mu_1 \wedge \mu_2)$, cela veut dire que $\mathcal{I}$ est un modèle de $\mu_1 \wedge \mu_2$ qui est le plus proche de $\mathsf{Mod}(\omega)$ dans l'ensemble $\mathsf{Mod}(\mu_1 \wedge \mu_2)$. Supposons que $\mathcal{I}$ n'est pas un modèle de $(\omega \circ \mu_1) \wedge \mu_2$, ce qui est contradictoire. En effet, la condition de **(R∘6)** dit qu'il existe une interprétation $\mathcal{J}$ qui est un modèle de $\omega \circ \mu_1$ et de μ_2 ; c'est-à-dire $\mathcal{J}$ est un modèle de $\mu_1 \wedge \mu_2$ qui est le plus proche de $\mathsf{Mod}(\omega)$ dans l'ensemble $\mathsf{Mod}(\mu_1)$. Cela implique que $\mathcal{J}$ est plus proche de $\mathsf{Mod}(\omega)$ que $\mathcal{I}$ dans l'ensemble $\mathsf{Mod}(\mu_1)$. Cependant, nous savons déjà que $\mathcal{I}$ est plus proche de $\mathsf{Mod}(\omega)$ que $\mathcal{J}$ dans l'ensemble $\mathsf{Mod}(\mu_1 \wedge \mu_2)$, d'où la contradiction. En résumé, **(R∘6)** dit que si $\mathcal{I}$ est la plus proche de $\mathsf{Mod}(\omega)$ dans l'ensemble $\mathsf{Mod}(\mu_1 \wedge \mu_2)$, alors $\mathcal{I}$ est aussi la plus proche de $\mathsf{Mod}(\omega)$ dans l'ensemble $\mathsf{Mod}(\mu_1)$.

Les auteurs [Katsuno and Mendelzon, 1991] ont aussi proposé un théorème qui montre

l'équivalence entre les six postulats et une stratégie de révision fondée sur les pré-ordres totaux. Le théorème est fondé sur la notion d'*affectation fidèle* ("*faithful assignment*" en anglais). Soit $\mathcal{M}$ l'ensemble de toutes les interprétations d'un langage L. Un pré-ordre $\leq$ sur $\mathcal{M}$ est une relation réflexive et transitive sur $\mathcal{M}$. La relation $<$ est définie comme suit : $\mathcal{I} < \mathcal{I}'$ si et seulement si $\mathcal{I} \leq \mathcal{I}'$ et $\mathcal{I}' \not\leq \mathcal{I}$ pour deux interprétations $\mathcal{I}, \mathcal{I}'$. Un pré-ordre est *total* si pour toutes $\mathcal{I}, \mathcal{J} \in \mathcal{M}$, soit $\mathcal{I} \leq \mathcal{J}$, soit $\mathcal{J} \leq \mathcal{I}$. Une fonction qui affecte chaque formule $\phi \in L$ à un pré-ordre total $\leq_\phi$ sur les interprétations $\mathcal{M}$ est appelée une *affectation fidèle* si et seulement si les conditions suivantes sont respectées.

1. Si $\mathcal{I}, \mathcal{I}' \in \mathsf{Mod}(\phi)$ alors $\mathcal{I} <_\phi \mathcal{I}'$ n'est pas satisfaite.
2. Si $\mathcal{I} \in \mathsf{Mod}(\phi)$ et $\mathcal{I}' \notin \mathsf{Mod}(\phi)$ alors $\mathcal{I} <_\phi \mathcal{I}'$ est satisfaite.
3. Si $\phi \equiv \mu$, alors $\leq_\phi = \leq_\mu$.

Ces conditions garantissent qu'un modèle de ϕ ne peut être strictement inférieur à tous les autres modèles de ϕ et doit être strictement inférieur à toutes les interprétations qui ne sont pas des modèles de ϕ.

Une fonction $min(\mathsf{Mod}(\mu), \leq_\phi)$ peut être définie comme suit : $min(\mathsf{Mod}(\mu), \leq_\phi) = \{\mathcal{I} \mid \mathcal{I} \in \mathsf{Mod}(\mu)$ et il n'existe aucune $\mathcal{I}' \in \mathsf{Mod}(\mu)$ telle que $\mathcal{I}' <_\phi \mathcal{I}\}$. Si $\mathcal{I}' <_\phi \mathcal{I}$ exprime que $\mathcal{I}'$ est plus proche de $\mathsf{Mod}(\phi)$ que $\mathcal{I}$, alors la fonction $min(\mathsf{Mod}(\mu), \leq_\phi)$ peut être vue comme l'ensemble de tous les modèles de μ qui sont les plus proches de $\mathsf{Mod}(\phi)$.

Théorème 1. *[Katsuno and Mendelzon, 1991] Un opérateur de révision $\circ$ satisfait les postulats **(R$\circ$1)**-**(R$\circ$6)** si et seulement s'il existe une affectation fidèle qui affecte chaque formule ϕ à un pré-ordre total $\leq_\phi$ telle que $\mathsf{Mod}(\phi \circ \mu) = min(\mathsf{Mod}(\mu), \leq_\phi)$.*

Le Théorème 1 confirme qu'un opérateur de révision satisfaisant les postulats d'AGM peut être construit en définissant une distance sur un ensemble d'interprétations telle que cette distance soit un pré-ordre total.

En pratique, les approches classiques de révision lors de la construction d'un opérateur de révision peuvent être classées en deux grandes catégories à savoir les approches fondées sur les *syntaxes (approches syntaxiques)* et celles fondées sur les *modèles (approches sémantiques)* [Eiter and Gottlob, 1992].

2.1 Approches syntaxiques

Les approches syntaxiques manipulent directement des entités syntaxiques telles que les formules d'une base de connaissances. Pour prendre en compte une nouvelle formule en préservant la cohérence, ces approches tentent d'identifier d'autres formules qui doivent être retirées de la base de connaissance. L'avantage principal des approches fondées sur les syntaxes est de permettre la distinction de la pertinence de différentes formules [Nebel, 1991]. Par exemple, nous pouvons affecter une priorité inférieure aux formules qui peuvent changer et une priorité supérieure à celles qui seraient "protégées". Les principales limites sont les suivantes :

(i) les procédures résultant de ces approches dépendent fortement de la syntaxe des bases de connaissances. Pour deux bases de connaissances différentes K_1 et K_2 qui ont la même signification, c'est-à-dire $Cn(K_1) = Cn(K_2)$, la révision peut conduire à deux résultats différents. Par exemple, soient $K_1 = \{\mu, \phi\}$ et $K_2 = \{\mu \wedge \phi\}$ qui sont équivalentes, on constate que la révision de K_1 et de K_2 par $\neg\mu$ donne des résultats différents. En effet, dans le cas de K_1, μ et ϕ sont deux formules distinctes, la révision de K par μ ne devrait donc pas produire un effet sur ϕ, et le résultat de révision est $\{\neg\mu, \phi\}$. Dans le deuxième cas où K_2 contient une seule formule $\mu \wedge \phi$, le fait que la négation de μ contredit cette formule, elle devrait donc être enlevée de la base de connaissances K_2. Donc, le résultat de la révision de K_2 par $\neg\mu$ devrait être $\{\neg\mu\}$;

(ii) le résultat de révision peut ne pas être *à grain fin* ("*fine-grained*" en anglais [Qi and Du, 2009]). Dans ce cas, une formule peut être supprimée complètement même si seulement une partie de celle-ci cause des incohérences. Considérons l'exemple ci-dessus dans le cas où K_2 est révisée par $\neg\mu$, la seule formule $\mu \wedge \phi$ de K_2 est supprimée même si ϕ ne contredit pas $\neg\mu$.

Malgré ces inconvénients, il existe quelques opérations de révision de croyances fondées sur les syntaxes développées pour la révision d'une ontologie en LD.

2.1.1 Révision par affaiblissement

Qi *et al.* [Qi *et al.*, 2006] ont reformulé les postulats **(R∘1)**-**(R∘6)** pour la révision d'ontologies en LD et proposé deux opérateurs de révision. Le premier opérateur est l'opérateur de révision fondé sur l'*affaiblissement* (*weakening* en anglais) qui est défini par l'affaiblissement des axiomes dans une base de connaissances en LD. Pour affaiblir un axiome contradictoire, cet opérateur de révision le supprime simplement et peut donner des résultats contre-intuitifs dans certains cas.

Dans le travail de Qi *et al.* [Qi *et al.*, 2006], une base de connaissances est définie en logique $\mathcal{ALC}$ avec nominaux $\mathcal{O}$ [Schaerf, 1994]. Sachant qu'un nominal a de la forme $\{a\}$ où a est un nom d'individu, la sémantique de $\{a\}$ est définie par $\{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}$ pour une interprétation $\mathcal{I}$. Soient K et K' deux bases de connaissances en LD, les postulats **(R∘1)**-**(R∘6)** sont reformulés en utilisant des ensembles de modèles $\mathsf{Mod}(K)$ et $\mathsf{Mod}(K')$, e.g.,

(G3) si K' est cohérente alors $\mathsf{Mod}(K \circ K') \neq \emptyset$;

(G4) si $\mathsf{Mod}(K_1) = \mathsf{Mod}(K_2)$ et $\mathsf{Mod}(K'_1) = \mathsf{Mod}(K'_2)$ alors $\mathsf{Mod}(K_1 \circ K'_1) = \mathsf{Mod}(K_2 \circ K'_2)$.

Le premier opérateur de révision est fondé sur l'affaiblissement d'une GCI et d'une assertion. Une GCI affaiblie $(C \sqsubseteq D)_{weak}$ d'une $C \sqsubseteq D$ est de la forme $(C \sqcap \neg\{a_1\} \neg \ldots \neg\{a_n\}) \sqsubseteq D$ où n est le nombre d'individus à supprimer de C. De plus, $d((C \sqsubseteq D)_{weak})$ est utilisé pour désigner le degré de $(C \sqsubseteq D)_{weak}$. Une assertion affaiblie ϕ_{weak} d'une assertion $\phi = C(a)$ est de la forme $\phi_{weak} = \top(a)$ où $\phi_{weak} = \phi$. Le degré de ϕ_{weak} est $d(\phi_{weak}) = 1$ si $\phi_{weak} = \top(a)$, $d(\phi_{weak}) = 0$ sinon. Une base de connaissances $K_{weak,K'}$ est une base de connaissances affaiblie de K par rapport à K' si elle satisfait (i) $K_{weak,K'} \cup K'$ est cohérente, et (ii) il existe une bijection f de K à $K_{weak,K'}$ telle que pour chaque $\phi \in K'$, $f(\phi)$ est une affaiblissement de ϕ.

L'ensemble de toutes les bases affaiblies de K par rapport à K' est désigné par $Weak_{K'}(K)$.
Exemple 2. Soit $K = \{bird(tweety), bird \sqsubseteq flies\}$ et $K' = \{\neg flies(tweety)\}$, où $bird$ et $flies$ sont deux concepts et $tweety$ est un nom d'individu. Il est facile de vérifier que $K \cup K'$ est incohérente. Dans ce cas, nous pouvons trouver deux bases affaiblies de K par rapport à K', à savoir $K_1 = \{\top(tweety), bird \sqsubseteq flies\}$ et $K_2 = \{bird(tweety), bird \sqcap \neg\{tweety\} \sqsubseteq$

$flies\}$. Donc, $Weak_{K'}(K) = \{K_1, K_2\}$.

Le degré $d(K_{weak,K'})$ est défini par $d(K_{weak,K'}) = \sum_{\phi \in K_{weak,K'}}(d(\phi))$. Et le résultat de révision fondée sur les affaiblissements d'une base de connaissances K par une autre K', désigné par $K \circ_w K'$, peut être défini comme suit :
$K \circ_w K' = K' \cup K_{weak,K'} \mid K_{weak,K'} \in Weak_{K'}(K)$ et

$$\nexists K_i \in Weak_{K'}(K), d(K_i) < d(K_{weak,K'}).$$

Pour capturer la sémantique du résultat de révision, les auteurs [Qi *et al.*, 2006] ont proposé une notation d'*exception*. Soient $\mathcal{W}$ un ensemble non-vide d'interprétations, $\mathcal{I} \in \mathcal{W}$, ϕ un axiome et K une base de connaissances, si ϕ est une assertion, le nombre de ϕ-exceptions $e^{\phi}(\mathcal{I})$ est 0 si $\mathcal{I}$ satisfait ϕ et 1 sinon. Si ϕ est une GCI de la forme $C \sqsubseteq D$, le nombre ϕ-exceptions pour $\mathcal{I}$ est $e^{\phi}(\mathcal{I}) = |C^{\mathcal{I}} \cap (\neg D^{\mathcal{I}})|$ si $C^{\mathcal{I}} \cap (\neg D^{\mathcal{I}})$ est fini, sinon $e^{\phi}(\mathcal{I}) = \infty$; et le nombre de K-exceptions pour $\mathcal{I}$ est $e^{K}(\mathcal{I}) = \sum_{\phi \in K} e^{\phi}(\mathcal{I})$. L'ordre $\prec_K$ sur $\mathcal{W}$ est $\mathcal{I} \prec_K \mathcal{I}'$ ssi $e^{K}(\mathcal{I}) \leq e^{K}(\mathcal{I}')$ pour $\mathcal{I}' \in \mathcal{W}$.

Avec les notations précédentes, les auteurs [Qi *et al.*, 2006] ont énoncé la proposition $\mathsf{Mod}(K \circ_w K') = min(\mathsf{Mod}(K'), \prec_K)$ disant que les modèles de résultat de révision sont ceux de K' qui sont minimaux par rapport à l'ordre $\prec_K$ induits par K. Ils ont aussi montré que leur opérateur de révision satisfait les postulats sauf le **(G4)**.

Dans l'exemple 2, le résultat de révision de K par K' est soit $K \circ_w K' = K' \cup K_1$, soit $K \circ_w K' = K' \cup K_2$ car $d(K_1) = d(K_2) = 1$.

L'exemple 3 montre que l'opérateur de révision $\circ_w$ n'est pas assez précis.

Exemple 3. Soient $K = \{bird \sqcap flies(tweety), bird(chirpy)\}$ et $K' = \{\neg flies(tweety)\}$. Clairement, $bird \sqcap flies(tweerty)$ contredit $\neg flies(tweety)$ dans K'. Si $\phi = bird \sqcap flies(tweerty)$ alors l'affaiblissement de ϕ est $\phi_{weak} = \top(tweety)$. Donc, pour affaiblir ϕ, on l'a supprimée. Cependant, la connaissance $bird(tweety)$ n'a causé aucune contradiction dans $K \cup K'$ mais elle est supprimée aussi avec ϕ.

La conséquence est qu'un autre opérateur de révision a été proposé pour affiner le premier. Cependant, les auteurs ont considéré uniquement un affinage de l'affaiblissement

des assertions dans l'ABox en redéfinissant l'affaiblissement des assertions où un concept est soit un concept atomique A ou $\neg A$, soit $\exists R.C ou \forall R.C, soit \{b\}$, soit une disjonction de ces concepts. Cet affaiblissement des assertions avec celui des GCI précédent sont ensuite utilisés pour définir un opérateur de révision $\circ_{rw}$ tel que $\mathsf{Mod}(K \circ_{rw}) \subseteq \mathsf{Mod}(K \circ_w)$. Cet opérateur de révision satisfait aussi les postulats sauf le **(G4)**.

En résumé, aucun de leurs opérateurs ne satisfait tous les postulats d'AGM. De plus, il est difficile d'implémenter ces opérateurs car on devrait détecter les GCIs et les assertions qui sont responsables de conflit entre deux bases de connaissances.

2.1.2 Autres révisions fondées sur les syntaxes

Qi *et al.* [Qi *et al.*, 2008] ont proposé un opérateur de révision de "*noyau*" (*"kernel"* en anglais) dans des ontologies en LD fondées sur *MIPS* ("minimal incoherence-preserving sub-terminologies" en anglais, qui correspond à des sous-terminologies minimales de conservation de l'incohérence) et une *fonction d'incision* ("incision function" en anglais). La notion de MIPS est similaire à la notion d'un *ensemble de noyaux* dans le changement de base de croyances défini par Hansson [Hansson, 1994]. Dans la logique classique, étant donnée une base de connaissances K, un ensemble de formules classiques et une formule ϕ, un ϕ-noyau de K est la sous-base minimale de K qui implique ϕ. Pour définir une fonction de contraction afin de supprimer des connaissances d'une base de connaissances, appelée contraction du noyau, Hansson a défini une fonction d'incision qui sélectionne les formules à supprimer dans chaque ϕ-noyau de K. Qi *et al.* ont adapté cette fonction dans la révision des TBoxes en sélectionnant des axiomes de chaque MIPS pour les supprimer de l'ontologie originelle. Les auteurs ont aussi développé deux algorithmes pour calculer l'opérateur de révision. Cependant, le premier, en général, est difficile à calculer car il a besoin de calculer tous les MIPS, tandis que le second peut supprimer trop d'axiomes de l'ontologie initiale après la révision (*i.e.* le résultat de la révision n'est pas "*à grain fin*").

Ribeiro et Wassermann [Ribeiro and Wassermann, 2007] ont étudié la contraction d'ontologie avec l'identité de Levy (*i.e.* retrait d'un axiome). L'absence de la négation d'axiome dans les LD a conduit les auteurs à étudier la *semi-révision* d'ontologie avec deux construc-

tions différentes pour la révision d'une base de connaissances K par un axiome α. La première construction assure que la base de connaissances résultante est toujours cohérente, mais α n'est pas nécessairement inféré à partir de cette dernière, *i.e.* le succès de la révision n'est pas garanti. Quant à la deuxième, le succès de la révision est garanti mais la cohérence de la base de connaissances résultante, elle, ne l'est pas. Par conséquent, les postulats concernant la réussite et la cohérence ne sont pas garantis simultanément.

Calvanese et ses collègues [Calvanese *et al.*, 2010] ont proposé un algorithme de révision qui, étant donné une base de connaissances K et une nouvelle connaissance $\mathcal{N}$, renvoie un sous-ensemble de la fermeture déductive $cl(K)$ tel qu'il soit un ensemble maximal compatible d'axiomes pour K et $\mathcal{N}$. Cependant, le résultat de la révision est non-déterministe en général car il peut exister plusieurs sous-ensembles de $cl(K)$ dont chacun est un ensemble maximal compatible d'axiomes par rapport à K et $\mathcal{N}$. Pour atteindre le déterminisme, il est nécessaire de restreindre l'opération de révision sur l'ABox comme l'ont souligné les auteurs [Calvanese *et al.*, 2005, 2006].

2.2 Approches sémantiques

Contrairement aux approches syntaxiques, les approches fondées sur les modèles manipulent des modèles d'ontologies plutôt que leurs entités syntaxiques. Nous allons introduire deux approches sémantiques qui sont largement adaptés à la révision d'une ontologie en LD. Pour mieux comprendre ces adaptations, nous allons présenter un travail existant qui a proposé un opérateur de révision fondé sur une structure, dite "feature", qui représente un modèle d'une ontologie en LD.

2.2.1 Opérateurs de révision spécifiques

Les principales problématiques des approches sémantiques sont liées à la définition d'une distance entre les modèles et à la façon de calculer le résultat de la révision à partir des modèles sélectionnés en fonction de la distance définie. Des opérateurs spécifiques de révision [Dalal, 1988; Satoh, 1988] ont été proposés sur la base de la distance entre des

modèles de la base de connaissances et de nouvelles connaissances. L'opérateur de révision de Dalal [Dalal, 1988] utilise la cardinalité pour mesurer la distance alors que celui de Satoh [Satoh, 1988] utilise l'ensemble d'inclusion ("containment set" en anglais) fondé sur la différence symétrique $(S \setminus S') \cup (S' \setminus S)$ pour deux ensembles S, S'.

Pour la révision d'une base de connaissances K par une nouvelle formule μ, l'opérateur de révision de Dalal [Dalal, 1988] définit d'abord la distance de Hamming, qui est une distance d'une interprétation à une autre, comme suit :

$$d(\mathcal{I}_1, \mathcal{I}_2) = \Sigma_{p \in P} |\mathcal{I}_1(p) - \mathcal{I}_2(p)|.$$

Ensuite, une distance d'une interprétation à une base de connaissances est définie comme suit :

$$d(\mathcal{I}_1, K) = min_{\mathcal{I}' \in \mathsf{Mod}(K)} d(\mathcal{I}, \mathcal{I}').$$

Cette dernière distance permet la définition d'un pré-ordre sur les modèles de l'information d'entrée $\mathcal{I}_1 \leq_K \mathcal{I}_2$ ssi $d(\mathcal{I}_1, K) \leq d(\mathcal{I}_2, K)$. Les interprétations les plus proches de la base de connaissances sont les modèles du résultat de révision $\mathsf{Mod}(K \circ_d \mu) = min(\mathsf{Mod}(\mu), \leq_K)$. Selon le théorème 1, l'opérateur de révision de Dalal satisfait les six postulats **(R∘1)**-**(R∘6)**.

Exemple 4. Soit L un langage formel qui contient quatre symboles a, b, c, d. Considérons les cinq interprétations suivantes de L :

$\mathcal{I}_1 = (1,1,1,1)$, $\mathcal{I}_2 = (0,0,0,0)$, $\mathcal{J}_1 = (1,1,0,0)$, $\mathcal{J}_2 = (0,1,1,1)$, $\mathcal{J}_3 = (1,1,1,0)$.

Soit K une base de connaissances dont l'ensemble de modèles est $\mathsf{Mod}(K) = \{\mathcal{I}_1, \mathcal{I}_2\}$, soient ϕ_1, ϕ_2 et ϕ_3 des formules où $\mathsf{Mod}(\phi_1) = \{\mathcal{J}_1, \mathcal{J}_2, \mathcal{J}_3\}$, $\mathsf{Mod}(\phi_2) = \{\mathcal{J}_1, \mathcal{J}_2\}$ et $\mathsf{Mod}(\phi_3) = \{\mathcal{J}_1, \mathcal{J}_3\}$,en calculant la distance entre des interprétations, on peut obtenir les résultats de révision suivants :

$\mathsf{Mod}(K \circ_d \phi_1) = \{\mathcal{J}_2, \mathcal{J}_3\}$ car $d(\mathcal{I}_1, \mathcal{J}_2) = d(\mathcal{I}_1, \mathcal{J}_3) = 1$ est minimale ;
$\mathsf{Mod}(K \circ_d \phi_2) = \{\mathcal{J}_2\}$ car $d(\mathcal{I}_1, \mathcal{J}_2) = 1$ est minimale ;
$\mathsf{Mod}(K \circ_d \phi_3) = \{\mathcal{J}_3\}$ car $d(\mathcal{I}_1, \mathcal{J}_3) = 1$ est minimale.

Pour l'opérateur de révision de Satoh [Satoh, 1988], la distance $d(\mathcal{I}, \mathcal{J})$ entre deux

interprétations propositionnelles $\mathcal{I}, \mathcal{J}$ est définie comme la différence symétrique qui est l'ensemble de symboles dont l'interprétation est différente dans $\mathcal{I}$ et $\mathcal{J}$. Pour la révision d'une base de connaissances K par une formule μ, la distance minimale entre K et μ est définie par $d_{min}(K, \mu) = min_{\subseteq}(\{d(\mathcal{I}, \mathcal{J}) \mid \mathcal{I} \in \mathsf{Mod}(K), \mathcal{J} \in \mathsf{Mod}(\mu)\})$; et les modèles du résultat de la révision sont définis $\mathsf{Mod}(K \circ_s \mu) = \{\mathcal{J} \in \mathsf{Mod}(\mu) \mid \exists \mathcal{I} \in \mathsf{Mod}(K)$ tel que $d(\mathcal{I}, \mathcal{J}) \in d_{min}(K, \mu)\}$. L'opérateur de révision de Satoh satisfait cinq postulats **(R∘1)**-**(R∘5)** mais ne satisfait pas le postulat **(R∘6)**.

Exemple 5. Reconsidérons l'exemple 4, les distances entre les interprétations sont : $d(\mathcal{I}_1, \mathcal{J}_1) = \{c, d\}$, $d(\mathcal{I}_1, \mathcal{J}_2) = \{a\}$, $d(\mathcal{I}_1, \mathcal{J}_3) = \{d\}$, $d(\mathcal{I}_2, \mathcal{J}_1) = \{a, b\}$, $d(\mathcal{I}_2, \mathcal{J}_2) = \{b, c, d\}$, $d(\mathcal{I}_2, \mathcal{J}_3) = \{a, b, c\}$.

Ensuite, nous avons :
$d_min(K, \phi_1) = \{\{a\}, \{d\}\}$, $d_min(K, \phi_2) = \{\{a\}, \{c, d\}\}$, $d_min(K, \phi_3) = \{\{d\}, \{a, b\}\}$;

ce qui rend possible les résultats de révision suivants :

$\mathsf{Mod}(K \circ_s \phi_1) = \{\mathcal{J}_2, \mathcal{J}_3\}$;
$\mathsf{Mod}(K \circ_s \phi_2) = \{\mathcal{J}_1, \mathcal{J}_2\} = \mathsf{Mod}(\phi_2)$;
$\mathsf{Mod}(K \circ_s \phi_3) = \{\mathcal{J}_1, \mathcal{J}_3\} = \mathsf{Mod}(\phi_3)$.

Supposons qu'il existe une affectation fidèle d'un pré-ordre $\leq_K$ qui capture l'opérateur $\circ_s$. Alors, $\mathsf{Mod}(K \circ_s \phi_1) = \{\mathcal{J}_2, \mathcal{J}_3\}$ implique soit que $\mathcal{J}_2 <_K \mathcal{J}_1$, soit que $\mathcal{J}_3 <_K \mathcal{J}_1$. Cependant, $\mathcal{J}_2 \not<_K \mathcal{J}_1$ car $\mathsf{Mod}(K \circ_s \phi_2) = \mathsf{Mod}(\phi_2)$, et $\mathcal{J}_3 \not<_K \mathcal{J}_1$ car $\mathsf{Mod}(K \circ_s \phi_3) = \mathsf{Mod}(\phi_3)$, ce qui est contradictoire ! Il peut donc ne pas exister une affectation fidèle d'un pré-ordre qui capture l'opérateur $\circ_s$ pour toutes les bases de connaissances, c'est-à-dire qu'il existe certains postulats que l'opérateur $\circ_s$ ne satisfait pas.

2.2.2 Adaptation à la révision d'ontologies en LD

Lors de l'adaptation des approches sémantiques à la révision d'une ontologie en LD, il existe aussi des problèmes qui peuvent survenir pendant le traitement des modèles d'ontologies en LD. Premièrement, les ontologies en LD peuvent avoir un nombre infini de

modèles qui rendent impossible la construction d'une ontologie résultant d'une révision à partir des modèles. Deuxièmement, les modèles d'une ontologie en LD ont généralement des structures complexes (éventuellement infinies), ce qui peut nécessiter une définition complexe de la distance entre deux modèles. Troisièmement, il peut ne pas exister une unique ontologie qui admette exactement un ensemble donné de modèles [Grau *et al.*, 2012; Kharlamov *et al.*, 2013].

En dépit de ces problèmes, il y a eu de nombreuses tentatives pour adapter les approches sémantiques de révision aux ontologies en LD.

Qi et Du [Qi and Du, 2009] ont adapté l'opérateur de révision de Dalal [Dalal, 1988] pour la révision des terminologies en LD. Leurs opérateurs qui ne dépendent pas d'une LD spécifique sont définis à l'aide des ensembles de modèles des bases de connaissances. Ils ont également montré que leurs opérateurs satisfont les postulats d'AGM. Cependant, les auteurs n'ont proposé aucune procédure de calcul d'une ontologie de révision.

Wang *et al.* [Wang *et al.*, 2010] ont adapté l'opérateur de révision de Satoh [Satoh, 1988] sur les ontologies en DL-Lite$^{\mathcal{N}}_{bool}$ [Artale *et al.*, 2007] qui étend les langages DL-Lite de base [Calvanese *et al.*, 2005] avec des opérateurs booléens complets et des restrictions de cardinalité dans ces bases de connaissances. Les auteurs ont introduit une structure finie, à savoir *feature*, pour représenter un modèle éventuellement infini pour les ontologies en DL-Lite. Bien que l'opérateur de révision introduit n'assure pas entièrement le principe du changement minimal, il permet aux auteurs de développer et de mettre en œuvre un algorithme de révision des ontologies DL-Lite. Une nouvelle notion appelée *sémantique de type* [Zhuang *et al.*, 2014; Wang *et al.*, 2015; Zhuang *et al.*, 2016] généralisée à partir de la notion de *features* a été développée plus tard pour fournir une méthode traitant la révision et la contraction sur les ontologies en DL-Lite. Comme la *sémantique de type* est fondée sur des modèles propositionnels plutôt que sur des structures de premier ordre, il n'est pas facile d'étendre cette sémantique à une LD plus expressive. Il semble donc que ce n'est pas la voie la plus adéquate pour étendre cette approche à la révision des ontologies en $\mathcal{SHIQ}$. Cependant, ces travaux nous suggèrent une façon de définir une opération de révision ainsi que construire l'ontologie révisée. Dans la section suivante, nous visualisons la révision avec *feature* pour donner une piste de cette approche.

2.2.3 Révision avec "*feature*"

Dans le travail de Wang *et al.* [Wang *et al.*, 2010] sur la révision des ontologies en DL-Lite$^{\mathcal{N}}_{bool}$, pour chaque ontologie, une signature $\mathcal{S} = \mathcal{S}_C \cup \mathcal{S}_R \cup \mathcal{S}_I \cup \mathcal{S}_N$ peut être construite où $\mathcal{S}_C$ est un ensemble de concepts atomiques, $\mathcal{S}_R$ est un ensemble de rôles atomiques, $\mathcal{S}_I$ est un ensemble de noms d'individu et $\mathcal{S}_N$ est un ensemble de nombres naturels. La syntaxe de DL-Lite$^{\mathcal{N}}_{bool}$ est définie par les règles suivantes :

$$R \leftarrow P \mid P^{-}$$
$$B \leftarrow \top \mid \bot \mid A \mid \geqslant n\ R$$
$$C \leftarrow B \mid \neg C \mid C_1 \sqcap C_2$$

où $n \in \mathcal{S}_N$, $A \in \mathcal{S}_C$, $P \in \mathcal{S}_R$. B est appelé un concept basique et C est un concept général.

Soit $\mathcal{T}$ une TBox et $\mathcal{A}$ une ABox en DL-Lite$^{\mathcal{N}}_{bool}$, une ontologie dans DL-Lite$^{\mathcal{N}}_{bool}$ est toujours représentée en tant qu'une ontologie $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$.

Feature. [Wang *et al.*, 2010] Une "*Feature*" pour DL-*Lite*$^{\mathcal{N}}_{bool}$ est fondée sur la notion de *types*. Un $\mathcal{S}$-*type* τ est un ensemble de concepts basiques sur $\mathcal{S}$ tel que $\top \in \tau$, et pour tout $m, n \in \mathcal{S}_N$ avec $m < n$, $\geqslant n\ R \in \tau$ implique $\geqslant m\ R \in \tau$. Par exemple, $\mathcal{S}_C = \{A, B\}$, $\mathcal{S}_R = \{P\}$ et $\mathcal{S}_N = \{1, 3\}$, alors $\tau = \{A, \exists P, \geqslant 3P, \exists P^{-}\}$ est un *type*.

La définition d'un *type* τ satisfaisant un concept est :

- τ satisfait le concept basique B si $B \in \tau$,
- τ satisfait $\neg C$ si τ ne satisfait pas C,
- τ satisfait $C_1 \sqcap C_2$ si τ satisfait les deux C_1 et C_2,
- τ satisfait une inclusion de concepts $C_1 \sqsubseteq C_2$ si τ satisfait $\neg C_1 \sqcup C_2$,
- τ satisfait une TBox $\mathcal{T}$ s'il satisfait toute inclusion dans $\mathcal{T}$.

Les *types* sont suffisants pour capturer la sémantique des TBoxes car ils ne se réfèrent pas à des individus mais ils sont insuffisants pour capturer la sémantique des ABoxes. Les auteurs [Wang *et al.*, 2010] ont donné une notion de types avec les individus à savoir un ensemble $\mathcal{S}$-Herbrand.

Un ensemble $\mathcal{S}$-Herbrand $\mathcal{H}$ est un ensemble fini d'assertions de la forme $B(a)$ ou $P(a, b)$, où $a, b \in \mathcal{S}_I$, $P \in \mathcal{S}_R$ et B est un concept basique sur $\mathcal{S}$ satisfaisant les conditions suivantes :

1. Pour chaque $a \in \mathcal{S}_I$, $\top(a) \in \mathcal{H}$, $\bot(a) \notin \mathcal{H}$, et $\geqslant n\ R(a) \in \mathcal{H}$ implique $\geqslant m\ R(b) \in \mathcal{H}$ pour $m, n \in \mathcal{S}_N$ avec $m < n$.
2. Pour chaque $P \in \mathcal{S}_R$, $P(a, b_i) \in \mathcal{H}$ $(i = 1, ..., n)$ implique $\geqslant m\ P(a) \in \mathcal{H}$ pour tout $m \in \mathcal{S}_N$ tel que $m \leq n$.
3. Pour chaque $P \in \mathcal{S}_R$, $P(b_i, a) \in \mathcal{H}$ $(i = 1, ..., n)$ implique $\geqslant m\ P^-(a) \in \mathcal{H}$ pour tout $m \in \mathcal{S}_N$ tel que $m \leq n$.

Soit $\mathcal{H}$ un ensemble Herbrand pour l'ontologie $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ et un individu a, le *type* de a dans $\mathcal{H}$ est $\tau(a, \mathcal{H}) = \{C |\ C(a) \in \mathcal{H}\}$. Les conditions 2 et 3 conservent la cohérence d'un ensemble Herbrand.

L'ensemble Herbrand $\mathcal{H}$ satisfait l'assertion de concept $C(a)$ si $\tau(a, \mathcal{H})$ satisfait le concept C. L'ensemble Herbrand $\mathcal{H}$ satisfait l'assertion de rôle $P(a, b)$ si $P(a, b)$ est dans $\mathcal{H}$ et $\neg P(a, b)$ si $P(a, b)$ n'est pas dans $\mathcal{H}$. L'ensemble Herbrand $\mathcal{H}$ satisfait une ABox $\mathcal{A}$ si $\mathcal{H}$ satisfait toutes les assertions dans $\mathcal{A}$.

Pour le raisonnement sur une ontologie, les auteurs ont argumenté que les types τ et les ensembles $\mathcal{H}$ caractérisent les modèles des ontologies et qu'on peut utiliser un couple $\langle \tau, \mathcal{H} \rangle$, où τ est un type et $\mathcal{H}$ est un ensemble Herbrand, au lieu des modèles standards. Cependant, ce couple est insuffisant pour vérifier la cohérence de l'ontologie. Par exemple, considérons l'ontologie $\mathcal{O} = \langle \{\exists P^- \sqsubseteq \bot\},\ \{\exists P(a)\} \rangle$. Évidemment, cette ontologie $\mathcal{O}$ est incohérente et n'a donc pas de modèle. Cependant, soit $\mathcal{S} = \{P, a, 1\}$, alors $\langle \tau, \mathcal{H} \rangle = \langle \{\exists P\},\ \{\exists P(a)\} \rangle$ satisfait $\mathcal{O}$.

Donc, la définition de "*feature*" est introduite afin de capturer la sémantique non seulement de la TBox mais aussi de la ABox d'une ontologie, comme suit :

Soit $\mathcal{S}$ une signature, une $\mathcal{S}$-"*feature*" est définie en tant qu'une paire $\mathcal{F} = \langle \Xi, \mathcal{H} \rangle$, où Ξ est un ensemble non vide de $\mathcal{S}$-*types* et $\mathcal{H}$ est un ensemble Herbrand, satisfaisant les conditions suivantes :

1. $\exists P \in \bigcup \Xi$ si et seulement si $\exists P^- \in \bigcup \Xi$, pour chaque $P \in \mathcal{S}_R$.

2. $\tau(a, \mathcal{H}) \in \Xi$, pour chaque $a \in \mathcal{S}_I$.

La première condition garantit que $\exists P$ est insatisfiable en fonction de la TBox si et seulement si $\exists P^-$ est aussi insatisfiable en fonction de la TBox. La seconde condition exige que si $\mathcal{F}$ satisfait l'assertion $C(a)$ alors C doit être satisfiable en fonction de la TBox.

Les auteurs [Wang *et al.*, 2010] ont défini la relation de satisfaction d'une inclusion et d'une assertion par rapport à une "*feature*" $\mathcal{F} = \langle \Xi, \mathcal{H} \rangle$ comme suit :

— $\mathcal{F}$ satisfait $C_1 \sqsubseteq C_2$ si τ satisfait $\neg C_1 \sqcup C_2$ pour $\forall \tau \in \Xi$.

— $\mathcal{F}$ satisfait $C(a)$ si le *type* de a dans $\mathcal{H}$ satisfait C.

— $\mathcal{F}$ satisfait $P(a, b)$ ou $P^-(b, a)$ si $P(a, b)$ est dans $\mathcal{H}$.

Par conséquent, $\mathcal{F}$ est appelée une "*feature*" de l'ontologie $\mathcal{O}$ si elle satisfait chaque inclusion et chaque assertion dans $\mathcal{O}$. $MF(\mathcal{O})$ est l'ensemble de toutes les "*features*" de $\mathcal{O}$.

Exemple 6. Considérons l'ontologie $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ introduite par les auteurs [Wang *et al.*, 2010], où

$$\mathcal{T} = \{A \sqsubseteq \exists P,\ B \sqsubseteq \exists P,\ \exists P^- \sqsubseteq B,\ A \sqcap B \sqsubseteq \bot,\ \geq 2P^- \sqsubseteq \bot\}$$
$$\mathcal{A} = \{A(a),\ P(a, b)\}$$

Soit $\mathcal{S} = Sig(\mathcal{O}) = \{A, B, P, 1, 2, a, b\}$. Alors $\mathcal{F} = \langle \Xi, \mathcal{H} \rangle$ est une "*feature*" (fini) de $\mathcal{O}$, où

$$\Xi = \{\tau_1,\ \tau_2\} \text{ avec } \tau_1 = \{A,\ \exists P\} \text{ et } \tau_2 = \{B,\ \exists P,\ \exists P^-\}, \text{ et}$$
$$\mathcal{H} = \{A(a),\ \exists P(a),\ B(b),\ \exists P(b),\ \exists P^-(b),\ P(a, b)\}.$$

Approximation maximale [De Giacomo *et al.*, 2007]. Cette notion est utilisée pour résoudre le problème d'inexpressibilité. Soit $\mathbb{M}$ un ensemble d'interprétations et $\mathcal{S}$ une signature ; dans la plupart des cas, il n'existe pas une ontologie $\mathcal{O}$ sur $\mathcal{S}$ telle que les modèles de $\mathcal{O}$ soient exactement $\mathbb{M}$. Et en l'appliquant aux "*features*", une ontologie $\mathcal{O}$ est appelée une *approximation maximale* d'un ensemble $\mathbb{F}$ de $\mathcal{S}$-*features* si (1) $Sig(\mathcal{O}) \subseteq \mathcal{S}$, (2) $\mathbb{F} \subseteq MF(\mathcal{O})$, et (3) il n'existe pas une ontologie $\mathcal{O}'$ sur $\mathcal{S}$ telle que $\mathbb{F} \subseteq MF(\mathcal{O}') \subset MF(\mathcal{O})$.

Opération de révision avec features Soient $\mathcal{F}_1 = \langle \Xi_1, \mathcal{H}_1 \rangle$ et $\mathcal{F}_2 = \langle \Xi_2, \mathcal{H}_2 \rangle$ deux $\mathcal{S}$-"*features*", la *distance* entre $\mathcal{F}_1$ et $\mathcal{F}_2$, $\mathcal{F}_1 \triangle \mathcal{F}_2$ est une paire $\langle \Xi_1 \triangle \Xi_2, \mathcal{H}_1 \triangle \mathcal{H}_2 \rangle$, où $X \triangle Y$ est la différence symétrique entre deux ensembles X et Y : $X \triangle Y = (X \cup Y) \setminus (X \cap Y)$.

Pour comparer deux distances, $\mathcal{F}_1 \triangle \mathcal{F}_2 \subseteq \mathcal{F}_3 \triangle \mathcal{F}_4$ si $\Xi_1 \triangle \Xi_2 \subseteq \Xi_3 \triangle \Xi_4$ et $\mathcal{H}_1 \triangle \mathcal{H}_2 \subseteq \mathcal{H}_3 \triangle \mathcal{H}_4$; et $\mathcal{F}_1 \triangle \mathcal{F}_2 \subset \mathcal{F}_3 \triangle \mathcal{F}_4$ si $\mathcal{F}_1 \triangle \mathcal{F}_2 \subseteq \mathcal{F}_3 \triangle \mathcal{F}_4$ et $\mathcal{F}_3 \triangle \mathcal{F}_4 \not\subseteq \mathcal{F}_1 \triangle \mathcal{F}_2$.

Soient $\mathcal{O}$, $\mathcal{O}'$ deux ontologies dans DL-$Lite^{\mathcal{N}}_{bool}$ et une signature $\mathcal{S} = Sig(\mathcal{O}, \mathcal{O}')$. La f-révision de $\mathcal{O}$ par $\mathcal{O}'$, $\mathcal{O} \circ_f \mathcal{O}'$ est telle que $MF(\mathcal{O} \circ_f \mathcal{O}') = MF(\mathcal{O}')$ si $MF(\mathcal{O}) = \emptyset$; sinon

$$MF(\mathcal{O} \circ_f \mathcal{O}') = \{\langle \Xi', \mathcal{H}' \rangle \in MF(\mathcal{O}') \mid \exists \langle \Xi, \mathcal{H} \rangle \in MF(\mathcal{O}) \ t.q \\ \mathcal{H} \triangle \mathcal{H}' \in d_H(\mathcal{O}, \mathcal{O}') \ et \ \langle \Xi \triangle \Xi', \mathcal{H} \triangle \mathcal{H}' \rangle \in d_F(\mathcal{O}, \mathcal{O}')\}.$$

où

$$d_H(\mathcal{O}, \mathcal{O}') = min_{\subseteq}(\ \{\mathcal{H} \triangle \mathcal{H}' \mid \exists \langle \Xi, \mathcal{H} \rangle \in MF(\mathcal{O}),\ \exists \langle \Xi', \mathcal{H}' \rangle \in MF(\mathcal{O}')\}\ ),$$
$$d_F(\mathcal{O}, \mathcal{O}') = min_{\subseteq}(\ \{\mathcal{F} \triangle \mathcal{F}' \mid \exists \mathcal{F} \in MF(\mathcal{O}),\ \exists \mathcal{F}' \in MF(\mathcal{O}')\}\ ).$$

L'ensemble de features du résultat de révision $MF(\mathcal{O} \circ_f \mathcal{O}')$ est un ensemble de features $\mathcal{F}' \in MF(\mathcal{O}')$ tel qu'il existe une feature $\mathcal{F} \in MF(\mathcal{O})$ et tel que la distance entre $\mathcal{F}$ et $\mathcal{F}'$ soit minimale.

Cet opérateur de révision satisfait les cinq premiers postulats suivants, reformulation de ceux d'AGM, e.g.,

(R5) $(\mathcal{O} \circ \mathcal{O}') \cup \mathcal{O}'' \models \mathcal{O} \circ (\mathcal{O}' \cup \mathcal{O}'')$;

(R6) si $(\mathcal{O} \circ \mathcal{O}') \cup \mathcal{O}''$ est cohérente alors $\mathcal{O} \circ (\mathcal{O}' \cup \mathcal{O}'') \models (\mathcal{O} \circ \mathcal{O}') \cup \mathcal{O}''$.

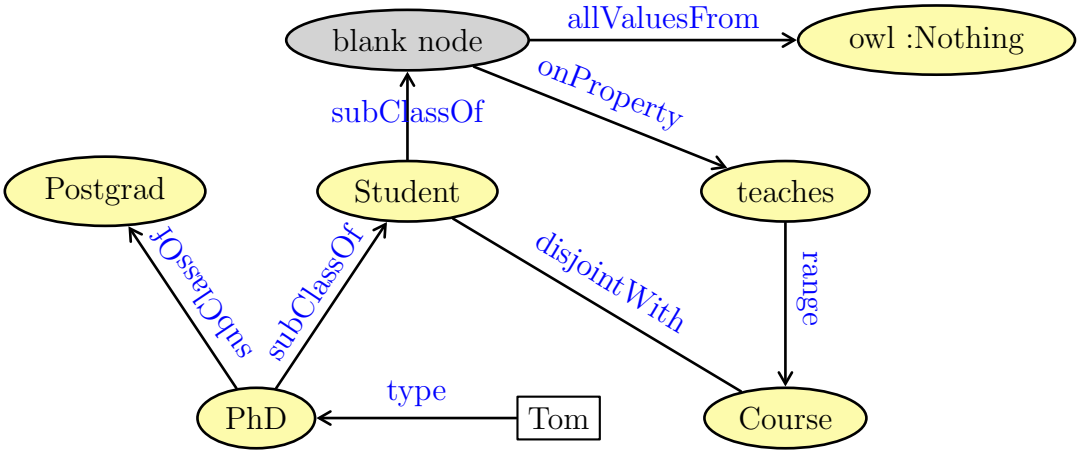
Le postulat **(R6)** n'est toujours pas satisfait dans certains cas.

Algorithme pour l'opération de révision. Dans le but de calculer le résultat de la révision d'une ontologie, les auteurs [Wang *et al.*, 2010] ont proposé un algorithme en se basant sur les définitions de feature et des opérateurs de révision. L'entrée de cet algorithme considère deux ontologies $\mathcal{O}$ et $\mathcal{O}'$ dans le langage DL-$Lite^{\mathcal{N}}_{bool}$ et $\mathcal{S} = Sig(\mathcal{O} \cup \mathcal{O}')$. La sortie générée est le résultat de révision $\mathcal{O} \circ_f \mathcal{O}'$. En premier lieu,

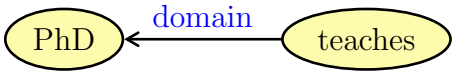
l'algorithme calcule les features $MF(\mathcal{O})$ et $MF(\mathcal{O}')$. Puis il calcule $MF(\mathcal{O} \circ_f \mathcal{O}')$ à partir de $MF(\mathcal{O})$ et $MF(\mathcal{O}')$ par la définition présentée précédemment. Ensuite, pour chaque τ qui n'existe pas dans l'ensemble de types dans $MF(\mathcal{O} \circ_f \mathcal{O}')$, l'algorithme ajoute l'inclusion $C_\tau \sqsubseteq \bot$ à $\mathcal{T}$ où $C_\tau = \bigsqcap_{B \in \tau} B \sqcap \bigsqcap_{B \notin \tau} \neg B$. Et pour chaque $a \in \mathcal{S}_I$, il ajoute l'assertion $(\bigsqcup_{\tau \in \Xi_a} C_\tau)(a)$ à $\mathcal{A}$ où $\Xi_a = \{\tau |\ \exists \langle \Xi, \mathcal{H} \rangle \in MF(\mathcal{O} \circ_f \mathcal{O}'\}$ tel que τ est le type de a dans $\mathcal{H}$; puis, il ajoute toutes les assertions de rôle $P(a, b)$ à $\mathcal{A}$ pour chaque $P(a, b)$ qui existe dans l'ensemble Herbrand dans $MF(\mathcal{O} \circ_f \mathcal{O}'$. Enfin, l'algorithme retourne $\langle \mathcal{T}, \mathcal{A} \rangle$ comme résultat de la révision $\mathcal{O} \circ_f \mathcal{O}'$.

Considérons l'exemple 7 des auteurs [Wang *et al.*, 2010] qui ont proposé une ontologie originelle $\mathcal{O}$ et une ontologie $\mathcal{O}'$ utilisée pour lancer la révision.

Exemple 7. $\mathcal{O} = \langle \{PhD \sqsubseteq Student \sqcap Postgrad,$
$Student \sqsubseteq \neg \exists teaches,\ \exists teaches^- \sqsubseteq Course,$
$Student \sqcap Course \sqsubseteq \bot\},\ \{PhD(Tom)\} \rangle$



$\mathcal{O}' = \langle \{PhD \sqsubseteq \exists teaches\},\ \emptyset \rangle$

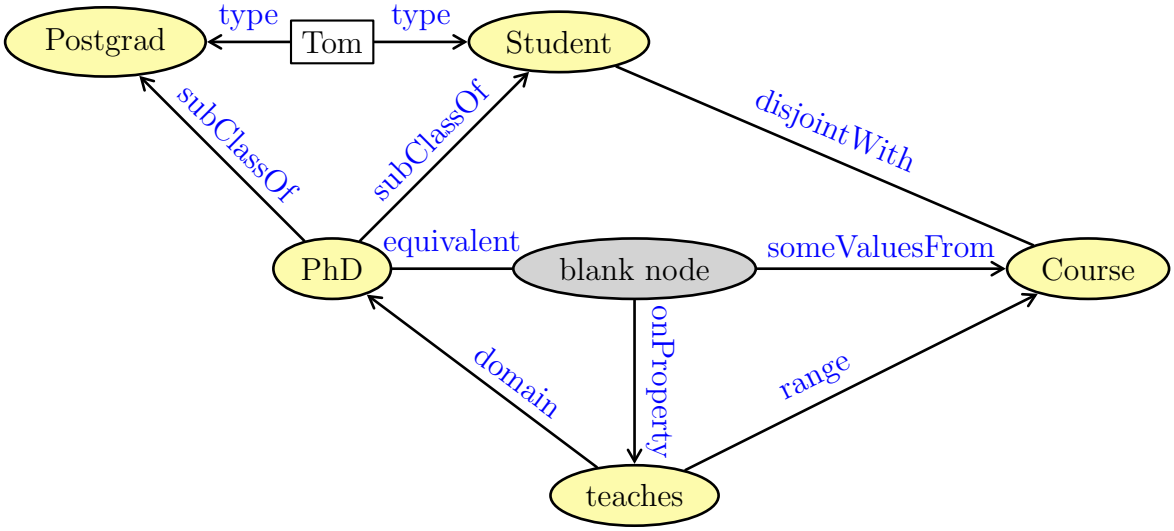


La TBox de $\mathcal{O}$ montre que les PhD sont les étudiants diplômés n'ayant pas le droit d'enseigner des cours. L'ABox montre que Tom est un PhD. D'autre part, la TBox de $\mathcal{O}'$ montre que les PhD ont maintenant le droit d'enseigner. Donc, le résultat de la révision est une *approximation maximale*.

$\mathcal{O} \circ_f \mathcal{O}' = \langle \{PhD \sqsubseteq Student \sqcap Postgrad,\ PhD \sqsubseteq \exists teaches,$

$Student \sqcap \exists teaches \sqsubseteq PhD,\ \exists teaches^{-} \sqsubseteq Course,$

$Student \sqcap Course \sqsubseteq \bot\},\ \{Student(Tom),\ Postgrad(Tom)\}\ \rangle.$



Les auteurs [Wang *et al.*, 2010] ont démontré que le résultat de la révision en utilisant cet algorithme retourne toujours l'approximation maximale de $\mathcal{O} \circ_f \mathcal{O}'$. Dans le pire des cas, cependant, la complexité de cet algorithme est exponentielle et la taille de l'ontologie révisée est exponentiellement bornée par la taille de l'ontologie originelle. Cet algorithme est optimisé par rapport à sa version première mais cette amélioration n'est pas radicale. Pour étendre ce travail, nous pouvons appliquer cette approche dans un langage plus expressif. Par exemple, nous pouvons ajouter l'inclusion de rôles $R \sqsubseteq S$ ou la transitivité des rôles. Nous pouvons aussi définir un autre *feature* qui soit optimisée.

En résumé, les auteurs [Wang *et al.*, 2010] ont donné une nouvelle notion de *feature* (fini) en se basant sur la notion de modèles (infinis) pour définir les opérateurs de révision. Cependant, ce travail ne s'applique qu'au langage DL-Lite$^{\mathcal{N}}_{bool}$. Notre problématique est donc de déterminer si l'on peut définir un autre opérateur de révision pour des langages plus expressifs en se fondant sur une autre structure finie comme l'est celle de graphe. Il existe des algorithmes (*e.g.*, l'algorithme de tableau) qui peuvent construire des graphes représentants une structure finie de modèles infinis pour des langages plus expressifs.

2.3 Combinaison d'approches syntaxiques et sémantiques

Afin de trouver un bon compromis entre les deux catégories d'approches exposées précédemment, une approche hybride a tenté d'intégrer les approches sémantiques dans les approches fondées sur les syntaxes. Un résultat qui établit des relations entre des approches fondées sur les syntaxes et celles fondées sur les modèles a été présenté par Qi et ses collègues [Qi *et al.*, 2015]. Ils ont prouvé qu'un opérateur de révision reposant sur la syntaxe peut être utilisé pour approximer les opérateurs de révision fondés sur un modèle dans DL-Lite$_{\mathcal{R}}$ proposés par Kharlamov et ses collègues [Kharlamov *et al.*, 2013]. En d'autres termes, le résultat de l'opérateur de révision basé sur les syntaxes est une "meilleure" approximation de deux opérateurs de révision basés sur les modèles qui n'ont pas encore eu de procédure de calcul. Les auteurs [Qi *et al.*, 2015] ont également proposé un algorithme basé sur les graphes pour calculer la révision d'ontologies en utilisant des techniques de base de graphes pour calculer la révision d'ontologies. Dans leur graphe, chaque nœud représente un concept basique ou un rôle basique à partir de la signature d'une ontologie DL-Lite, et chaque arc représente une assertion dans l'ontologie.

2.4 Conclusion

Nous avons présenté deux catégories d'approches pour la révision des ontologies en LD. Une synthèse des approches de révision d'ontologies en LD est réalisée par Qi et Yang [Qi and Yang, 2008]. Nous trouvons que le résultat obtenu par des approches syntaxiques peut être trop éloigné de l'ontologie initiale, *i.e.* un axiome est supprimé complètement même si seulement une partie de celui-ci est la cause des incohérences. De plus, la procédure pour chercher des axiomes incohérents n'est pas pratique pour de grandes ontologies. C'est pourquoi nous avons fait le choix d'investiguer les approches sémantiques pour définir un opérateur de révision dans des logiques expressives en se fondant sur une structure finie comme celles d'arbre. Nous avons vu que l'algorithme de tableau construit des arbres représentants une structure finie de modèles infinis dans les logiques expressives comme $\mathcal{ALC}$

ou $\mathcal{SHIQ}$. Nous allons maintenant aborder plus en détail le cas de la révision d'ontologie d'expressivité $\mathcal{SHIQ}$.

Deuxième partie

Révision en Logique de Description expressive

Chapitre 3

Révision d'ontologies d'expressivité $\mathcal{SHIQ}$

Nous allons présenter dans ce chapitre une nouvelle approche fondée sur les modèles pour réviser des ontologies représentées en $\mathcal{SHIQ}$ qui est une LD expressive. Afin de simplifier la construction d'un opérateur de révision d'ontologie en $\mathcal{SHIQ}$, en première approche, nous considérerons une ontologie en $\mathcal{SHIQ}$ qui ne contient pas d'ABox. Une telle ontologie est une paire $(\mathcal{T}, \mathcal{R})$ (*cf.* section 1.2 du chapitre 1).

Pour résoudre les problèmes liés à la révision d'ontologies vus dans la section 2.2 du chapitre 2, nous utilisons un ensemble fini de structures finies, à savoir un *arbre de complétion*, pour représenter un ensemble de modèles éventuellement infini d'une ontologie en $\mathcal{SHIQ}$. Ces structures finies qui ont un pré-ordre total permettent de déterminer la différence sémantique entre deux ontologies représentées en tant que deux ensembles de modèles. En effet, la révision d'une ontologie $\mathcal{O}$ par une autre ontologie $\mathcal{O}'$ peut être réduite à la sélection des modèles "appropriés" à partir de l'ensemble de tous les modèles admis par $\mathcal{O}'$ tels que les modèles sélectionnés soient les plus proches des modèles de $\mathcal{O}$. Les modèles sélectionnés sont ensuite utilisés pour construire une ontologie résultante de la révision de $\mathcal{O}$ par $\mathcal{O}'$. Pour illustrer cette idée, considérons l'exemple 8.

Exemple 8. Soit $\mathcal{O}$ une ontologie contenant les axiomes :

$$\top \sqsubseteq \mathsf{BakerVideo} \sqcup \mathsf{PastryVideo} \sqcup \mathsf{User} \tag{3.1}$$

$$\mathsf{BakerVideo} \sqsubseteq \neg\mathsf{PastryVideo} \tag{3.2}$$

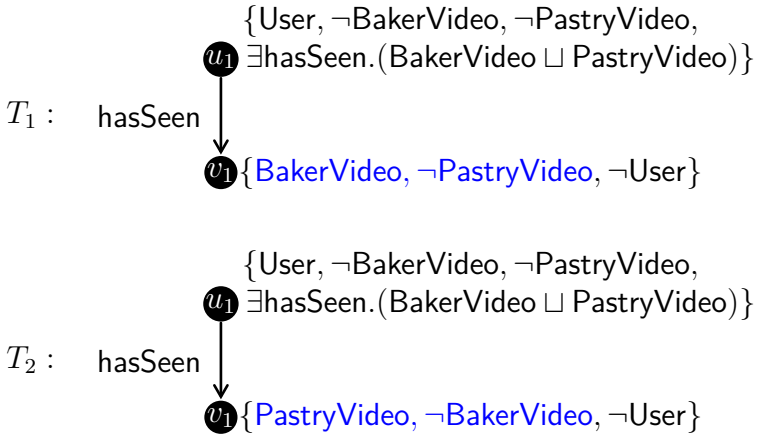
$$\mathsf{User} \sqsubseteq \neg\mathsf{BakerVideo} \sqcup \neg\mathsf{PastryVideo} \tag{3.3}$$

$$\mathsf{User} \sqsubseteq \exists\mathsf{hasSeen}.(\mathsf{BakerVideo} \sqcup \mathsf{PastryVideo}) \tag{3.4}$$

L'axiome 3.1 dit que "tout est une instance de la classe BakerVideo ou PastryVideo ou User". Les concepts BakerVideo et PastryVideo sont disjoints (*cf.* axiome 3.2), de même User et chaque BakerVideo et PastryVideo (*cf.* axiome 3.3). L'axiome 3.4 dit qu'"une instance de User doit avoir vu (hasSeen) une vidéo de BakerVideo ou de PastryVideo".
Une meilleure compréhension du domaine de la boulangerie-pâtisserie peut nous conduire à ajouter l'axiome suivant provenant d'une autre ontologie $\mathcal{O}'$:

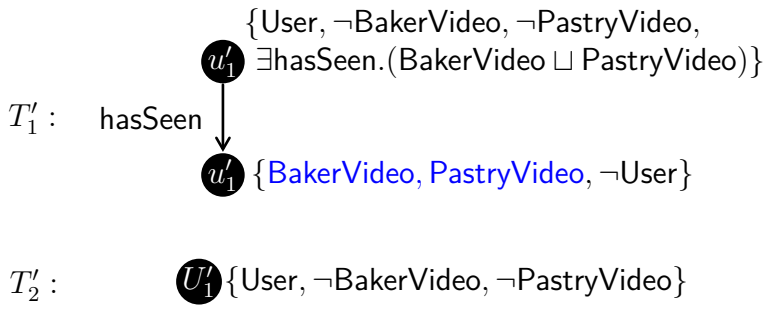
$$\top \sqsubseteq (\mathsf{BakerVideo} \sqcap \mathsf{PastryVideo}) \sqcup \mathsf{User} \tag{3.5}$$

Ce nouvel axiome stipule que "tout est une instance soit de BakerVideo et PastryVideo, soit de User" (*cf.* axiome 3.5). L'ontologie $\mathcal{O} \cup \mathcal{O}'$ est incohérente car l'axiome 3.5 contredit l'axiome 3.2. Rappelons que notre but est de construire une nouvelle ontologie $\mathcal{O}^*$ qui soit "compatible" avec les axiomes provenant de $\mathcal{O}'$ telle que $\mathcal{O}^*$ soit sémantiquement la plus proche de $\mathcal{O}$ (*i.e.* correspondant à un changement minimal). Nous pouvons vérifier que les deux arbres de complétion T_1 et T_2 suivants produisent des modèles de $\mathcal{O}$.



De même, nous pouvons vérifier que les deux arbres T'_1, T'_2 suivants produisent des modèles

de $\mathcal{O}'$.



En définissant une distance entre des arbres de complétion fondée sur la similarité structurelle, il est plausible de dire que T'_1 est plus proche de T_1 et T_2 que T'_2. Donc, une ontologie résultante $\mathcal{O}^*$ devrait admettre T'_1 plutôt que T'_2.

Une autre problématique à résoudre dans notre approche est la possibilité de la non existence d'une ontologie de révision telle qu'elle puisse être exprimée dans la logique utilisée pour l'expression des ontologies initiales $\mathcal{O}, \mathcal{O}'$, et qu'elle admette *exactement* un ensemble d'arbres de complétion comme ses modèles. Pour cette raison, nous empruntons des travaux de De Giacomo et ses collègues la notion d'*approximation maximale* [De Giacomo *et al.*, 2007] nous permettant de générer une ontologie sémantiquement minimale d'une révision telle qu'elle admette un ensemble de modèles construit à partir de $\mathcal{O}$ et $\mathcal{O}'$.

3.1 Notions liminaires

Dans cette section, nous décrirons quelques notions utiles pour la construction d'une nouvelle version de l'algorithme de tableau dans la section suivante.

Définition 1 (Sous-concept). *Soit $\mathcal{O} = (\mathcal{T}, \mathcal{R})$ une ontologie en $\mathcal{SHIQ}$ avec $\mathcal{S}(\mathcal{O}) = \mathbf{R} \cup \mathbf{C}$. Un ensemble $\mathsf{sub}(\mathcal{O})$ est défini de manière inductive comme suit :*

$\mathsf{sub}(\mathcal{O}) = \mathsf{sub}(\mathcal{T}) \cup \{\dot{\neg}C \mid C \in \mathsf{sub}(\mathcal{T})\}$;

$\mathsf{sub}(\mathcal{T}) = \bigcup_{C \sqsubseteq D \in \mathcal{T}} \mathsf{sub}(\mathsf{nnf}(\neg C \sqcup D))$;

$$\mathsf{sub}(C) = \begin{cases} \{C, \neg C\} & si\ C \in \mathbf{C} \\ \mathsf{sub}(E) \cup \mathsf{sub}(F) & si\ C \in \{E \sqcap F, E \sqcup F\} \\ \{C\} \cup \{\exists R'.E \mid R \sqsubseteq R'\} \cup \mathsf{sub}(E) & si\ C = \exists R.E \\ \{C\} \cup \{\forall R'.E \mid R \sqsubseteq R'\} \cup \mathsf{sub}(E) & si\ C = \forall R.E \\ \{C\} \cup \{\geq nR'.E \mid R \sqsubseteq R'\} \cup \mathsf{sub}(E) & si\ C = (\geq nR.E) \\ \{C\} \cup \mathsf{sub}(E) & si\ C = (\leq nR.E) \end{cases}$$

◁

Notons que tous les concepts exprimés sous la forme de disjonction et de conjonction sont remplacés par leurs disjoints et conjoints. De plus, $\mathsf{sub}(\mathcal{O})$ inclut (i) tous les sous-concepts se trouvant dans $\mathcal{O}$, (ii) des concepts sous la forme $\exists R'.C$, (resp. $\forall R'.C$ et $(\geq nR'.E)$) si $\exists R.C$ (resp. $\forall R.C$, $(\geq nR.E)$) se trouvant dans $\mathcal{O}$ avec $R \sqsubseteq R'$, et (iii) la négation de chaque concept ajoutée dans $\mathsf{sub}(\mathcal{O})$. Par exemple, l'ensemble de tous les sous-concepts dans l'ontologie $\mathcal{O}$ de l'Exemple 8 est $\mathsf{sub}(\mathcal{O}) = \{\mathsf{User}, \neg\mathsf{User}, \mathsf{BakerVideo}, \neg\mathsf{BakerVideo}, \mathsf{PastryVideo}, \neg\mathsf{PastryVideo}, \exists\mathsf{hasSeen}.(\mathsf{BakerVideo} \sqcup \mathsf{PastryVideo}), \forall\mathsf{hasSeen}.(\neg\mathsf{BakerVideo} \sqcap \neg\mathsf{PastryVideo})\}$.

Afin d'éviter le traitement des conjonctions et disjonctions au plus haut niveau d'un concept C (*i.e.* ceux qui n'apparaissent pas dans les restrictions (universelles, existentielles, ou de cardinalité) se trouvant dans C) lors de la construction d'un arbre de complétion pour une ontologie, nous utilisons une fonction $\mathsf{Flat}(C)$ (Flat pour le terme *flattening* en anglais) qui retourne un ensemble de sous-ensembles de $\mathsf{sub}(C)$.

Définition 2 (**Aplatissement**). *Soit C un $\mathcal{SHIQ}$-concept, nous définissons une fonction $\mathsf{Flat}(C)$ qui retourne un ensemble de sous-ensembles de $\mathsf{sub}(C)$ comme suit :*

1. *si C est un nom de concept ou C est une restriction existentielle, universelle ou de cardinalité alors $\mathsf{Flat}(C) = \{\{C\}\}$;*
2. *si $C = E \sqcup F$ alors $\mathsf{Flat}(C) = \mathsf{Flat}(E) \cup \mathsf{Flat}(F)$;*
3. *si $C = E \sqcap F$ alors $\mathsf{Flat}(C) = \{W \cup W' \mid W \in \mathsf{Flat}(E), W' \in \mathsf{Flat}(F)\}$.*

◁

L'application du point 3 dans la Définition 2 à un concept C peut augmenter $\mathsf{Flat}(C)$ exponentiellement. Par exemple, $\mathsf{Flat}((A_1 \sqcup B_1) \sqcap (A_2 \sqcup B_2)) = \{W \cup W' \mid W \in$

$\{\{A_1\},\{B_1\}\}, W' \in \{\{A_2\},\{B_2\}\}\} = \{\{A_1, A_2\},\{A_1, B_2\},\{B_1, A_2\},\{B_1, B_2\}\}$. Plus général, si $C = (A_1 \sqcup B1) \sqcap \cdots \sqcap (A_n \sqcup B_n)$, $\mathsf{Flat}(C)$ contient 2^n éléments. Nous formulons et prouvons certains propriétés utiles de la fonction Flat dans le lemme suivant.

Lemme 1. *Soit C un $\mathcal{SHIQ}$-concept alors :*

i. chaque élément $X \in \mathsf{Flat}(C)$ est un sous-ensemble de $\mathsf{sub}(C)$ et ne contient aucune conjonction ni aucune disjonction au plus haut niveau ;

ii. si C n'est ni une conjonction ni une disjonction alors $\mathsf{Flat}(C)$ contient un sous-ensemble unique de $\mathsf{sub}(C)$ qui n'inclut que C ;

iii. si C est une conjonction et un conjoint D de C n'est ni une conjonction ni une disjonction alors D doit apparaître dans tous les éléments de $\mathsf{Flat}(C)$;

iv. si C est une disjonction et chaque disjoint de C n'est ni une conjonction ni une disjonction alors tous les disjoints de C doivent apparaître dans les éléments distincts de $\mathsf{Flat}(C)$.

Démonstration.

i. Par les items 2 et 3 de la définition 2, chaque élément $X \in \mathsf{Flat}(C)$ ne contient aucune conjonction ni aucune disjonction au plus haut niveau. Selon la définition 1 et la définition 2, chaque élément $X \in \mathsf{Flat}(C)$ est un sous-ensemble de $\mathsf{sub}(C)$.

ii. Supposons que C n'est ni une conjonction ni une disjonction. Cela signifie que soit C est atomique, soit $C = \exists R.D$, soit $C = \forall R.D$, soit $C = (\leq n.r.D)$, soit $C = (\geq n.r.D)$. Par l'item 1 de la définition 2, nous avons $\mathsf{Flat}(C) = \{\{C\}\}$.

iii. Supposons que $C = D_1 \sqcap D_2 \sqcap \cdots \sqcap D_n$ et il existe un D_i $(1 \leq i \leq n)$ qui n'est ni une conjonction ni une disjonction. Sans réduction du caractère général de cette équation, supposons que $i = 1$ et réécrivons $C = D_1 \sqcap X$ avec $X = D_2 \sqcap \cdots \sqcap D_n$. En appliquant l'item 3 de la définition 2 à C, nous obtenons $\mathsf{Flat}(C) = \{\{D_1\} \cup W \mid W \in \mathsf{Flat}(X)\}$. Ceci implique que D_1 apparaît dans tous les éléments de $\mathsf{Flat}(C)$.

iv. Supposons que $C = D_1 \sqcup D_2 \sqcup \cdots \sqcup D_n$ et chaque disjonction D_i $(1 \leq i \leq n)$ de C n'est ni une conjonction ni une disjonction. Selon l'item 2 de la définition 2, nous avons $\mathsf{Flat}(C) = \mathsf{Flat}(D_1) \cup \cdots \cup \mathsf{Flat}(D_n) = \{\{D_1\}\} \cup \cdots \cup \{\{D_n\}\} = \{\{D_1\}, \ldots, \{D_n\}\}$. Donc, chaque élément de $\mathsf{Flat}(C)$ contient exactement une disjonction de C.

□

3.2 Caractérisation de la sémantique d'ontologie

Une approche sémantique pour la révision d'une ontologie devrait modifier les contraintes sémantiques de l'ontologie en manipulant l'ensemble de ses modèles. Cependant, l'ensemble de tous les modèles d'une ontologie peut être infini ; et, il existe des ontologies en $\mathcal{SHIQ}$ qui n'admettent que des modèles infinis. Ceci nous amène à proposer une méthode permettant de caractériser la sémantique d'une ontologie en $\mathcal{SHIQ}$ en utilisant un ensemble fini de *graphes de complétion*. Horrocks et ses collègues [Horrocks *et al.*, 1999] ont proposé un algorithme de tableau pour la vérification de la cohérence d'une ontologie $\mathcal{O}$ en $\mathcal{SHIQ}$. Cet algorithme tente de construire un graphe étiqueté fini, à savoir un *arbre de complétion*, à partir duquel un modèle peut être inventé pour $\mathcal{O}$. L'algorithme retourne "OUI" si un tel arbre de complétion est construit, et "NON" s'il ne construit pas un tel arbre de complétion après avoir considéré tous les cas possibles de non-déterminisme.

Dans cette section, nous présentons un algorithme avec de nouvelles règles nous permettant de construire l'ensemble de tous les arbres de complétion. Chacun de ceux-ci peut être utilisé pour inventer un modèle pour $\mathcal{O}$.

Définition 3 (**Arbre de complétion**). *Soit $\mathcal{O} = (\mathcal{T}, \mathcal{R})$ une ontologie en $\mathcal{SHIQ}$. Un* arbre de complétion *pour $\mathcal{O}$ est un arbre $T = \langle V, E, L, \hat{x}_i \rangle$ où*

• *V est un ensemble de nœuds contenant un nœud racine $\hat{x} \in V$. Chaque nœud $x \in V$ est étiqueté avec une fonction L telle que $L(x) \subseteq \mathsf{sub}(\mathcal{O})$. De plus, $\neq$ est une relation binaire symétrique sur V. E est un ensemble d'arêtes. Chaque arête $\langle x, y \rangle \in E$ est étiquetée avec un ensemble $L(\langle x, y \rangle)$ qui contient $\mathcal{SHIQ}$-rôles (éventuellement inverses) se trouvant dans $\mathcal{O}$.*

• *Si deux nœuds x et y sont connectés par une arête $\langle x, y \rangle$, alors y est dit* successeur *de x, désigné par $y \in \mathsf{succ}(x)$, et x est dit* prédécesseur *de y ; l'*ancêtre *est la fermeture transitive de* prédécesseur. *Un nœud y est un R-successeur de x si, pour chaque rôle R' avec $R' \sqsubseteq R$,*

Algorithme 1 : Algorithme de tableau construisant un arbre de complétion pour une ontologie en $\mathcal{SHIQ}$

Entrée : $\mathcal{O}$: une ontologie en $\mathcal{SHIQ}$
Sortie : Est-ce que $\mathcal{O}$ est cohérente ?

1 Soit $T = (V, L, E, \widehat{x})$ un arbre initial tel que $V = \{\widehat{x}\}$ et $L(\widehat{x}) = \emptyset$;
2 **tant que** *il y a une règle r dans Figure 3.1 qui peut être appliquée à un nœud* $x \in V$ **faire**
3 Appliquer r ;
4 **si** *il y a un arbre* T' *non-clash et complet qui est construit par Lignes 1 à 3* **alors**
5 **retourner** *OUI* ;
6 **retourner** *NON* ;

$R' \in L(\langle x, y\rangle)$; *x est un R-prédécesseur de y si y est un R-successeur de x. Un nœud y est un R-voisin de x si y est un R-successeur de x ou x est un* $\mathsf{Inv}(R)$*-successeur de y.*

• *Un nœud x est dit* bloqué *par y s'il a des ancêtres* x', y *et* y' *tels que (i) x est un successeur de* x' *et y est un successeur de* y'*, (ii)* $L(x) = L(y)$*,* $L(x') = L(y')$*, et (iii)* $L(\langle x', x\rangle) = L(\langle y', y\rangle)$.

• T *contient un* clash *s'il existe un nœud* $x \in V$ *tel que (i) soit* $\{A, \neg A\} \subseteq L(x)$ *pour chaque nom de concept* $A \in \mathbf{C}$*, (ii) soit* $(\leq nS.C) \in L(x)$ *et il y a des* $(n+1)$ S*-voisins* $y_1, \cdots, y_{n+1}$ *de x avec* $y_i \neq y_j$ *et* $X \subseteq L(y_i)$ *pour certain* $X \in \mathsf{Flat}(C)$ *et tout* $1 \leq i < j \leq (n+1)$. ◁

3.2.1 Nouvel algorithme de tableau

En se fondant sur le travail de Horrocks, Sattler et Tobies [Horrocks *et al.*, 1999], nous concevons un nouvel algorithme de tableau (*cf.* Algorithme 1) afin de construire un arbre de complétion en utilisant les règles d'expansion de la figure 3.1. Les règles sont décrites comme suit :

— Si la $\exists$-règle est applicable à un concept $\exists S.C$ dans l'étiquette $L(x)$ d'un nœud x qui n'est pas bloqué alors l'algorithme crée un nouveau S-voisin y de x et ajoute dans $L(y)$ un ensemble de concepts "aplati" de C par $\mathsf{Flat}(C)$.

— Si la $\forall$-règle est applicable à un concept $\forall S.C$ dans l'étiquette $L(x)$ d'un nœud x où

$\exists$-règle : si 1. $\exists S.C \in L(x)$, x n'est pas bloqué, et
2. il n'y a pas un S-voisin y de x tel que $X \subseteq L(y)$ pour certain $X \in \mathsf{Flat}(C)$
alors créer un nouveau nœud y avec $L(\langle x, y\rangle) := \{S\}$ et $L(y) := X$
pour certain $X \in \mathsf{Flat}(C)$.

$\forall$-règle : si 1. $\forall S.C \in L(x)$, et
2. il y a un S-voisin y de x tel que $X \not\subseteq L(y)$ pour tout $X \in \mathsf{Flat}(C)$
alors $L(y) := L(y) \cup X$ pour certain $X \in \mathsf{Flat}(C)$.

$\forall_+$-règle : si 1. $\forall S.C \in L(x)$,
2. il y a un R avec $\mathsf{Trans}(R)$ et $R \sqsubseteq S$, et
3. il y a un R-voisin y de x tel que $\forall R.C \notin L(y)$
alors $L(y) := L(y) \cup \{\forall R.C\}$.

$\geq$-règle : si 1. $(\geq nS.C) \in L(x)$, x n'est pas bloqué, et
2. il n'y a pas n S-voisins $y_1, \cdots, y_n$ de x tel que $X \subseteq L(y_i)$ pour certain
$X \in \mathsf{Flat}(C)$ et $y_i \neq y_j$ pour $0 \leq i < j \leq n$
alors créer n nouveaux nœuds $y_1, \cdots, y_n$ avec $L(\langle x, y_i\rangle) := \{S\}$,
$L(y_i) := X$ pour certain $X \in \mathsf{Flat}(C)$ et $y_i \neq y_j$ pour $1 \leq i < j \leq n$.

$\leq$-règle : si 1. $(\leq nS.C) \in L(x)$,
2. il y a $n+1$ S-voisins $y_0, \ldots, y_n$ de x s.t. $X \subseteq L(y_i)$
pour certain $X \in \mathsf{Flat}(C)$,
3. il y a deux S-voisins y, z de x avec $X_1 \subseteq L(y)$, $X_2 \subseteq L(z)$ pour
certain $X_1, X_2 \in \mathsf{Flat}(C)$, y n'est pas un ancêtre de z, et pas de $y \neq z$
alors (i) $L(z) := L(z) \cup L(y)$ et $L(\langle x, y\rangle) := \emptyset$
(ii) si z est un ancêtre de x
alors $L(\langle z, x\rangle) := L(\langle z, nx\rangle) \cup \{\mathsf{Inv}(R) \mid R \in L(\langle x, y\rangle)\}$;
sinon $L(\langle x, z\rangle) := L(\langle x, z\rangle) \cup L(\langle x, y\rangle)$, et
(iii) ajouter $u \neq z$ pour tout u tel que $u \neq y$.

sat-règle : si la sat-règle n'a jamais été appliquée à x
alors choisir un sous-ensemble $S \subseteq \mathsf{sub}(\mathcal{O})$ tel que
$$L(x) \cup \bigcup_{X \in \mathsf{Flat}(\mathsf{nnf}(\neg C \sqcup D)), C \sqsubseteq D \in \mathcal{T}} X \subseteq S, \text{ et}$$
faire $L(x) := S \cup \bar{S}$ où $\bar{S} = \{\doteq C \mid C \in \mathsf{sub}(\mathcal{O}) \setminus S\}$

FIGURE 3.1 – Règles d'expansion pour $\mathcal{SHIQ}$

y est un S-voisin de x tel que $X \not\subseteq L(y)$ pour tout $X \in \mathsf{Flat}(C)$ alors l'algorithme ajoute dans $L(y)$ un ensemble de concepts "aplati" de C par $\mathsf{Flat}(C)$.

— Si la $\forall_+$-règle est applicable à un concept $\forall S.C$ dans l'étiquette $L(x)$ d'un nœud x où y est un R-voisin de x tel que $\forall R.C \notin L(y)$ avec $\mathsf{Trans}(R)$ et $R \sqsubseteq S$ alors l'algorithme ajoute le concept $\forall R.C$ dans $L(y)$.

— Si la $\geq$-règle est applicable à un concept $\geq nS.C$ dans l'étiquette $L(x)$ d'un nœud x qui n'est pas bloqué, où il n'y a pas n S-voisins $y_1, \cdots, y_n$ de x tels que $X \subseteq L(y_i)$ pour des $X \in \mathsf{Flat}(C)$ et $y_i \neq y_j$ pour $0 \leq i < j \leq n$ alors l'algorithme crée n S voisins $y_1, \cdots, y_n$ de x et ajoute dans chaque $L(y_i)$ un ensemble de concepts "aplati" de C par $\mathsf{Flat}(C)$.

— Si la $\geq$-règle est applicable à un concept $\geq nS.C$ dans l'étiquette $L(x)$ d'un nœud x alors l'algorithme fusionne des étiquettes de deux S-voisins de x de telle façon qu'une étiquette de rôle entre x et un de ces deux voisins soit devenue vide.

— L'algorithme applique la sat-règle à un nœud x en choisissant un ensemble $S \subseteq \mathsf{sub}(\mathcal{O})$ et ajoutant S dans $L(x)$ tel que S couvre $L(x)$ existante et un sous-ensemble des concepts "aplatis" de $\neg C \sqcup D$ pour chaque axiome $C \sqsubseteq D$ dans $\mathcal{O}$.

Remarquons deux différences principales entre les règles de la figure 3.1 et celles de l'algorithme de tableau standard à savoir (i) l'absence des règles de conjonction et disjonction. Selon le lemme 1, l'application de la fonction Flat à un concept ajouté à l'étiquette d'un nœud supprime toutes les conjonctions et disjonctions au plus haut niveau de ce concept, (ii) la présence de la sat-règle (sat représente la notion de *saturation*). Le choix d'un sous-ensemble $S \subseteq \mathsf{sub}(\mathcal{O})$ dans la sat-règle doit inclure tous les concepts aplatis des axiomes GCI. De plus, la sat-règle ajoute dans chaque étiquette de nœud soit C soit $\dot{\neg}C$ pour chaque $C \in \mathsf{sub}(\mathcal{O})$. Cet état de fait peut conduire à une explosion exponentielle mais c'est nécessaire pour construire une approximation d'ontologie à partir d'un ensemble d'arbres de complétion et un ensemble de sous-concepts.

Pour une ontologie $\mathcal{O} = (\mathcal{T}, \mathcal{R})$, notre algorithme commence par initialiser un arbre T contenant un nœud racine et applique la règle sat-règle à ce nœud. L'algorithme applique les règles de la figure 3.1 à chaque nœud jusqu'à ce qu'aucune règle ne leur soit applicable.

Dans ce cas, T est *complet*. Si aucun de ses nœuds contient un clash, T est *non-clash*. Les applications des règles de génération ($\exists$- et $\geq$-règles) peuvent créer de nouveaux nœuds dont l'étiquette est entièrement remplie par une application de la sat-règle à chacun, ou partiellement remplie par l'une des $\forall$-, $\forall_+$-règles. Après avoir appliqué la sat-règle à un nœud, son étiquette n'est plus modifié. Ceci explique pourquoi les règles habituelles d'un algorithme de tableau classique telles que $\sqsubseteq$- et ch-règles deviennent inutiles avec l'application de la sat-règle. De plus, la procédure Flat rend les $\sqcap$- et $\sqcup$-règles non nécessaires car le comportement de ces règles est entièrement intégré dans Flat qui est non-déterministe. Une conséquence directe de la définition de la sat-règle est que son comportement est très non-déterministe vu que le nombre de possibilités pour choisir un sous-ensemble de sub est borné par une fonction exponentielle de la cardinalité de $\mathsf{sub}(\mathcal{O})$.

Lemme 2. ***(Terminaison, Correction et Complétude).***

Soit $\mathcal{O}$ une ontologie en $\mathcal{SHIQ}$.

1. *L'algorithme 1 se termine.*
2. *Si l'algorithme 1 peut être appliqué à $\mathcal{O}$ de telle façon qu'il construise un arbre de complétion non-clash et complet alors $\mathcal{O}$ est cohérente ;*
3. *si $\mathcal{O}$ est cohérente alors l'algorithme 1 peut être appliqué à $\mathcal{O}$ de telle façon qu'il construise un arbre de complétion non-clash et complet.*

Résumé de démonstration. La terminaison est une conséquence de la condition de blocage et de la monotonicité de la construction d'un arbre de complétion T, c'est-à-dire que l'algorithme ne supprime jamais rien de T. Pour prouver la correction, on peut concevoir un modèle à partir d'un arbre de complétion T en "dénouage" ("unraveling" anglais). Ce processus génère des descendants de nœuds bloqués en répliquant les descendants de nœuds de blocage. Le modèle obtenu est un arbre qui peut être une structure infinie. Pour démontrer la complétude, nous utilisons un modèle de $\mathcal{O}$ pour guider les règles d'expansion non-déterministes et choisir des concepts corrects (ou successeurs corrects) afin de construire un arbre de complétion non-clash et complet T. Une preuve complète peut être trouvée en annexe. □

Notons que l'algorithme de tableau peut construire un arbre de complétion dont la profondeur est limitée par une fonction exponentielle dans la taille de $\mathcal{O}$ en raison de la condition de blocage. Cela implique que la taille d'un arbre de complétion est limitée par une fonction doublement exponentielle de la taille de $\mathcal{O}$. Puisque l'algorithme crée un arbre de complétion d'une manière non-déterministe, il s'exécute en temps non-déterministe doublement exponentiel.

3.2.2 Modèle d'arbre

Soit $\mathcal{O}$ une ontologie en $\mathcal{SHIQ}$, un arbre de complétion non-clash et complet T construit en appliquant l'algorithme de tableau avec les règles d'expansion de la figure 3.1 à $\mathcal{O}$ s'appelle un *modèle d'arbre*. Selon le lemme 2, pour chaque arbre de complétion non-clash et complet T, il est possible d'obtenir un modèle désigné par $\mathcal{I}(T)$, par "dénouage" ("unraveling"). Dans ce cas, un nœud de T peut être reproduit pour un nombre infini d'individus du modèle. Étant donné un axiome $C \sqsubseteq D$, nous définissons $\mathcal{I}(T) \models (C \sqsubseteq D)$ si $C^{\mathcal{I}(T)} \subseteq D^{\mathcal{I}(T)}$.

Inversement, selon la complétude de l'algorithme de tableau (*cf.* Lemme 2), un arbre de complétion non-clash et complet T peut être construit à partir d'un modèle $\mathcal{I} \in \mathsf{Mod}(\mathcal{O})$, désigné par $T(\mathcal{I})$.

Contrairement à l'algorithme de tableau classique qui se termine lorsque un arbre de complétion est construit avec succès, ce nouvel algorithme de tableau doit considérer tous les cas non-déterministes et construire tous les modèles d'arbre pour $\mathcal{O}$.

Notation 1 ($\mathsf{MT}(\mathcal{O}, \mathsf{sub})$)**.** *Nous utilisons* $\mathsf{MT}(\mathcal{O})$ *pour désigner l'ensemble de tous les modèles d'arbre pour une ontologie* $\mathcal{O}$ *en* $\mathcal{SHIQ}$*. Nous pouvons alors étendre* $\mathsf{MT}(\mathcal{O})$ *à* $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ *comme suit. L'ensemble* $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ *est construit par le nouvel algorithme de tableau pour* $\mathcal{O}$ *avec un ensemble supplémentaire de concepts* $\mathsf{sub}(\mathcal{O}')$ *importé dans* $\mathsf{sub}(\mathcal{O})$ *lors de l'application de la* sat*-règle. Ceci permet d'importer des concepts supplémentaires dans des étiquettes de nœud d'un arbre de complétion pour* $\mathcal{O}$ *tout en respectant les axiomes de* $\mathcal{O}$*. Notons que nous n'importons aucune contrainte sémantique à partir de* $\mathcal{O}'$ *aux modèles d'arbre dans* $\mathsf{MT}(\mathcal{O})$ *lors de la construction de* $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$.

Ce qui est réellement réalisée dans cette construction est l'importation dans $\mathcal{O}$ de la signature de $\mathcal{O}'$ avec les formules écrites dans cette signature. Cette importation peut changer $\mathsf{MT}(\mathcal{O})$ *mais ne peut jamais changer la cohérence de $\mathcal{O}$.*

Nous utilisons maintenant les notations récemment introduites pour formuler et prouver les propriétés suivantes sur $\mathsf{MT}(\mathcal{O})$ qui caractérisent la sémantique d'une ontologie $\mathcal{O}$.

Corollaire 1. *Soient $\mathcal{O}$ et $\mathcal{O}'$ deux ontologies cohérentes en $\mathcal{SHIQ}$, alors :*

1. *étant donné α un axiome écrit en $\mathcal{S}(\mathcal{O})$, $\mathcal{I}(T) \models \alpha$ pour chaque modèle d'arbre $T \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\alpha))$ ssi $\mathcal{I} \models \alpha$ pour chaque modèle $\mathcal{I} \in \mathsf{Mod}(\mathcal{O})$;*
2. $\mathsf{MT}(\mathcal{O} \cup \mathcal{O}') = \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$.

Démonstration.

1. Supposons que pour chaque $T \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\alpha))$ nous ayons $\mathcal{I}(T) \models \alpha$ (*). Soit $\mathcal{I} \in \mathsf{Mod}(\mathcal{O})$ avec un domaine d'interprétation Δ. Montrons que $\mathcal{I} \models \alpha$. Pour chaque individu $a \in \Delta$ et pour chaque $C \in \mathsf{sub}(\alpha)$, nous avons $a \in C^{\mathcal{I}}$ ou $a \in (\neg C)^{\mathcal{I}}$. Nous pouvons modifier la construction de $T(\mathcal{I})$ à partir de $\mathcal{I}$ lors de la démonstration de complétude de l'algorithme de tableau de telle façon que pour chaque nœud x de $T(\mathcal{I})$ avec $\pi(x) = a$ (π est une fonction des nœuds de $\mathcal{F}(\mathcal{I})$ à Δ) et pour chaque $C \in \mathsf{sub}(\alpha)$ nous ajoutons C ou $\neg C$ à $L(x)$ si $a \in C^{\mathcal{I}}$ ou $a \in (\neg C)^{\mathcal{I}}$ respectivement. Nous désignons $T(\mathcal{I})$ modifié par $T'(\mathcal{I})$. Selon la définition de $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\alpha))$, nous avons $T'(\mathcal{I}) \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\alpha))$. Par l'hypothèse (*), nous obtenons $\mathcal{I}(T'(\mathcal{I})) \models \alpha$ (**).

   Soit $\alpha = (C \sqsubseteq D)$. Comme $\mathcal{I} \models C \sqsubseteq D$ ssi $\mathcal{O} \cup \{\top \sqsubseteq C \sqcap \neg D\}$ incohérente, montrons que $\mathcal{O} \cup \{\top \sqsubseteq C \sqcap \neg D\}$ est incohérente. Supposons que $\mathcal{O} \cup \{\top \sqsubseteq C \sqcap \neg D\}$ est cohérente. Ceci implique qu'il existe un modèle $\mathcal{J} \in \mathsf{Mod}(\mathcal{O} \cup \{\top \sqsubseteq C \sqcap \neg D\})$, *i.e.* $d \in ((C \sqcap \neg D))^{\mathcal{J}}$ pour chaque $d \in \Delta^{\mathcal{J}}$. Par la construction de $T'(\mathcal{J})$ à partir de $\mathcal{J}$, nous avons $(C \sqcap \neg D) \in L(x)$ où x est un nœud dans T' qui représente d. Par la construction de $\mathcal{I}(T'(\mathcal{J}))$ à partir de $T'(\mathcal{J})$, nous avons $d \in (C \sqcap \neg D)^{\mathcal{I}(T'(\mathcal{J}))}$ (i).

   D'autre part, nous avons $\mathcal{J} \in \mathsf{Mod}(\mathcal{O})$ car $\mathcal{J}$ satisfait une ontologie plus grande que $\mathcal{O}$. Selon (**), nous obtenons $\mathcal{I}(T'(\mathcal{J})) \models (C \sqsubseteq D)$, *i.e.* $C^{\mathcal{I}(T'(\mathcal{J}))} \subseteq D^{\mathcal{I}(T'(\mathcal{J}))}$, ce qui contredit (i).

Inversement, supposons que pour chaque $\mathcal{I} \in \mathsf{Mod}(\mathcal{O})$ nous avons $\mathcal{I} \models \alpha$. Soit $T \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\alpha))$. Démontrons que $\mathcal{I}(T) \models \alpha$. Ceci est simple puisque $\mathcal{I}(T) \in \mathsf{Mod}(\mathcal{O})$.

2. Tout d'abord, démontrons que $\mathsf{MT}(\mathcal{O} \cup \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$. Soit $T \in \mathsf{MT}(\mathcal{O} \cup \mathcal{O}')$. Comme T est construit en appliquant l'algorithme de tableau sur $\mathcal{O} \cup \mathcal{O}'$, chaque axiome dans $\mathcal{O} \cup \mathcal{O}'$ est satisfait dans chaque étiquette de nœud et d'arc de T, et la sat-règle fonctionne sur $\mathsf{sub}(\mathcal{O} \cup \mathcal{O}')$. Ceci implique que nous pouvons reconstruire T en appliquant l'algorithme de tableau sur $\mathcal{O}$ avec la sat-règle fonctionnant sur $\mathsf{sub}(\mathcal{O} \cup \mathcal{O}')$; et donc que $T \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$. De la même manière, $T \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$. Donc, $T \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$. Alors, $\mathsf{MT}(\mathcal{O} \cup \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ (i).

   Ensuite, démontrons que $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \subseteq \mathsf{MT}(\mathcal{O} \cup \mathcal{O}')$. Soit $T \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$. Comme $T \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$, chaque axiome dans $\mathcal{O}$ et $\mathcal{O}'$ est satisfait dans chaque étiquette de nœud et d'arc de T, et la sat-règle fonctionne sur $\mathsf{sub}(\mathcal{O} \cup \mathcal{O}')$. Ceci implique que nous pouvons reconstruire T en appliquant l'algorithme de tableau sur $\mathcal{O} \cup \mathcal{O}'$ avec la sat-règle fonctionnant sur $\mathsf{sub}(\mathcal{O} \cup \mathcal{O}')$. Ceci veut dire que $T \in \mathsf{MT}(\mathcal{O} \cup \mathcal{O}')$. Donc, $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \subseteq \mathsf{MT}(\mathcal{O} \cup \mathcal{O}')$ (ii).

   Selon (i) et (ii), nous avons $\mathsf{MT}(\mathcal{O} \cup \mathcal{O}') = \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$.

□

Le corollaire 1 affirme l'équivalence sémantique entre $\mathsf{Mod}(\mathcal{O})$ et $\mathsf{MT}(\mathcal{O})$. Ce résultat nous permet de remplacer un ensemble éventuellement infini $\mathsf{Mod}(\mathcal{O})$ par un ensemble fini $\mathsf{MT}(\mathcal{O})$ grâce aux constructions présentées dans les sections suivantes.

3.3 Opération de révision

L'objectif principal de la présente section est de définir une opération de révision qui permette de réviser une ontologie cohérente $\mathcal{O}$ par des axiomes d'une autre ontologie cohérente $\mathcal{O}'$, mais où $\mathcal{O} \cup \mathcal{O}'$ est incohérente. Une telle opération de révision renvoie un ensemble d'arbres de complétion qu'une ontologie de révision devrait admettre pour

prendre en compte les nouvelles connaissances de $\mathcal{O}'$, et être sémantiquement aussi proche que possible de $\mathcal{O}$. Ces propriétés sur l'opération de révision sont capturées par les postulats d'AGM reformulés pour les ontologies en LD [Qi and Du, 2009]. Pour atteindre cet objectif, nous devons introduire une distance entre deux arbres de complétion, ce qui donne un pré-ordre total sur eux et permet de parler de similarité entre deux ontologies. Notons que cette distance est définie pour deux arbres de complétion qui sont isomorphes.

3.3.1 Distance entre des modèles d'arbre

Définissons tout d'abord la notion d'isomorphisme entre deux arbres de complétion.

Définition 4 (Isomorphisme). *Soient $T = \langle V, L, E, \widehat{x}\rangle$ et $T' = \langle V', L', E', \widehat{x}'\rangle$ deux arbres de complétion.*

• *T et T' sont* isomorphes *s'il y a une bijection π de V à V' telle que $\pi(\widehat{x}) = \widehat{x}'$, et pour chaque $x \in V$ avec $\pi(x) = y$, on a $\pi(\mathsf{succ}(x)) = \mathsf{succ}(y)$. Dans ce cas, π est un isomorphisme entre T et T'.*

• *T et T' sont* équivalents *s'il y a un isomorphisme π entre T et T' tel que $L(x) = L'(\pi(x))$ pour chaque $x \in V$, et $L(\langle x, y\rangle) = L'(\langle \pi(x), \pi(y)\rangle)$ pour chaque $\langle x, y\rangle \in E$.* ◁

Notons qu'un isomorphisme peut toujours être obtenu entre deux arbres de complétion en ajoutant des nœuds et arêtes vides dans les arbres de complétion car les étiquettes de nœud et d'arête sont ignorées dans la définition d'isomorphisme. Grâce à ceci, un successeur d'un nœud x dans $\mathsf{succ}(x)$ peut toujours être associé à un successeur d'un nœud y dans $\mathsf{succ}(y)$ pour avoir $\pi(\mathsf{succ}(x)) = \mathsf{succ}(y)$ si $\pi(x) = y$. De plus, une distance entre deux arbres isomorphes peut être définie en étendant la définition de la différence symétrique $\triangle$. Rappelons que $S \triangle S' = (S \cup S') \setminus (S \cap S')$ pour les deux ensembles S et S'.

Définition 5 (Distance). *Soient $T = \langle V, L, E, \widehat{x}\rangle$ et $T' = \langle V', L', E', \widehat{x}'\rangle$ deux arbres de complétion, soit $\Pi(T, T')$ l'ensemble de tous les isomorphismes entre T et T', une distance entre T et T', notée par $T \triangle T'$, est définie comme suit :*

$$T \triangle T' = \mathsf{min}_{\pi \in \Pi(T,T')}\{\mathsf{max}_{x \in V}(|L(x) \triangle L'(\pi(x))|) \quad + \\ \mathsf{max}_{\langle x,y\rangle \in E}(|L(\langle x, y\rangle) \triangle L'(\langle \pi(x), \pi(y)\rangle)|)\}$$ ◁

La fonction max dans la définition 5 (distance) retourne la plus grande différence des étiquettes de nœud (ou celles d'arête) à partir de toutes les paires de nœuds $x, \pi(x)$ avec $x \in V$ (ou les paires d'arêtes $\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle$ avec $\langle x, y\rangle \in E$) pour des isomorphismes π entre deux arbres. Si max est enlevée selon la définition 5 (distance), une distance nulle n'implique pas l'identité des deux arbres en question. Nous pouvons vérifier que $\triangle$ est une distance sur un ensemble d'arbres isomorphes. En effet, (**symmetry**) $T \triangle T' = T' \triangle T$ est dû à la commutativité de l'opérateur $S \triangle S'$; (**identity**) $T \triangle T' = 0$ si et seulement si $T = T'$ est une conséquence du fait que $S \triangle S' = \emptyset$ ssi $S = S'$ pour tous les ensembles S, S'. (**triangle inequality**) $T \triangle T'' \leq (T \triangle T') + (T' \triangle T'')$ est une conséquence du fait que $S \triangle S'' \subseteq (S \triangle S') \cup (S' \triangle S'')$ pour tout ensemble S, S', S''. De plus, nous pouvons montrer que cette distance produit un pré-ordre total sur un ensemble d'arbres isomorphes. Pour cela, nous définissons une relation "$T \leq T'$" sur un ensemble d'arbres isomorphes, incluant un arbre T_0 contenant seulement des étiquettes vides, comme suit : $T \leq T'$ si $T_0 \triangle T \leq T_0 \triangle T'$.

Lemme 3. *La relation "$\leq$" est un pré-ordre total sur un ensemble d'arbres isomorphes.*

Démonstration. Soient T, T', T'' et T_0, des modèles d'arbre dans un ensemble d'arbres isomorphes où T_0 contient des étiquettes vides. Prouvons les propriétés suivantes :

1. **Transitivité :** par la définition de la relation "$\leq$", $T \leq T'$ implique $T_0 \triangle T \leq T_0 \triangle T'$, et $T' \leq T''$ implique $T_0 \triangle T' \leq T_0 \triangle T''$. Ainsi, $T_0 \triangle T \leq T_0 \triangle T''$, et donc $T \leq T''$. Par conséquent, si $T \leq T'$ et $T' \leq T''$ alors $T \leq T''$.
2. **Totalité :** comme $\triangle$ est une distance, nous avons donc $T_0 \triangle T \leq T_0 \triangle T'$ ou $T_0 \triangle T' \leq T_0 \triangle T$. Ceci implique $T \leq T'$ ou $T' \leq T$.
3. **Réflexivité :** comme $\triangle$ est une distance, nous avons $T_0 \triangle T \leq T_0 \triangle T$. Donc, $T \leq T$.

□

Toutes les notions ci-dessus fournissent suffisamment d'éléments pour définir une opération de révision pour la révision d'une ontologie $\mathcal{O}$ en $\mathcal{SHIQ}$ par une autre $\mathcal{O}'$. Cette opération détermine un ensemble de modèles que l'ontologie de révision doit admettre. Dans

l'hypothèse où la sémantique d'une ontologie $\mathcal{O}$ est caractérisée par $\mathsf{Mod}(\mathcal{O})$, le corollaire 1 nous permet de représenter la sémantique de $\mathcal{O}$ en utilisant un ensemble fini $\mathsf{MT}(\mathcal{O})$ au lieu d'un ensemble éventuellement infini $\mathsf{Mod}(\mathcal{O})$.

3.3.2 Nouvelle opération de révision

Définition 6 (**Opération de révision**). *Soient $\mathcal{O}$ et $\mathcal{O}'$, deux ontologies cohérentes en $\mathcal{SHIQ}$. Un ensemble de modèles d'arbre de la révision de $\mathcal{O}$ par $\mathcal{O}'$, désigné par $\mathsf{MT}(\mathcal{O},\mathcal{O}')$, est défini comme suit :*

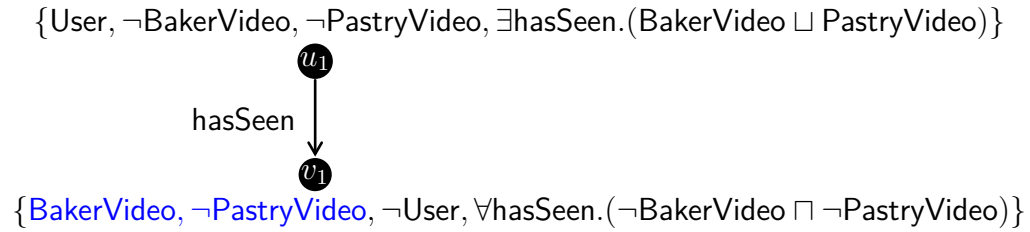
$$\mathsf{MT}(\mathcal{O},\mathcal{O}') = \{T \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \mid \exists T_0 \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')),$$

$$\forall T' \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O})), T'' \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) : T \vartriangle T_0 \leq T' \vartriangle T''\} \qquad \lhd$$

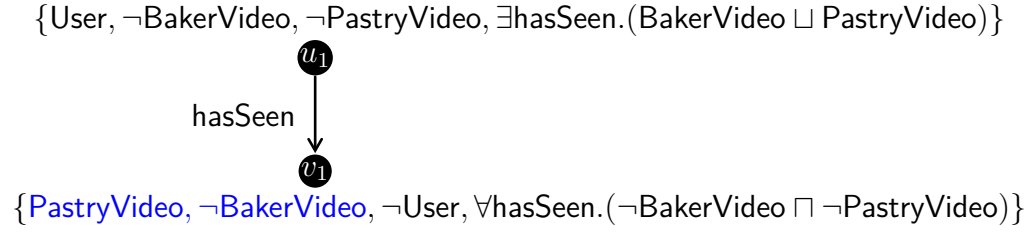
Intuitivement, parmi les modèles d'arbre dans $\mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$, $\mathsf{MT}(\mathcal{O},\mathcal{O}')$ ne retient que ceux qui sont les plus proches des modèles d'arbre de $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ grâce à l'opérateur $T_1 \vartriangle T_2$ qui caractérise la différence entre T_1 et T_2.

Exemple 9. Reconsidérons l'exemple 8. En appliquant le nouvel algorithme de tableau à $\mathcal{O}$, l'ensemble $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ contient les deux modèles d'arbre T_1 et T_2 suivants :

T_1 :



T_2 :



De même, en appliquant le nouvel algorithme de tableau à $\mathcal{O}$, nous obtenons l'ensemble $\mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ incluant les deux modèles d'arbre T_1' et T_2' suivants :

Selon la définition de notre distance, nous avons $T_1' \triangle T_1 = T_1' \triangle T_2 = T_2' \triangle T_1 = T_2' \triangle T_2 = 2$ qui sont minimales. Donc, $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$ contient deux modèles d'arbre T_1' et T_2'.

3.3.3 Satisfaction des Postulats d'AGM

Comme mentionné dans l'introduction générale, notre but est de proposer une opération de révision assurant le principe de changement minimal introduit par Alchourrón, Gärdenfors et Makinson [Alchourrón *et al.*, 1985] en tant que postulats dans le cadre de la révision de croyance. Katsuno et Mendelzon [Katsuno and Mendelzon, 1991] ont reformulé ces postulats pour des bases de connaissances propositionnelles, à savoir **(R∘1)**-**(R∘6)**, et montré que l'existence d'un pré-ordre total sur des modèles d'une base de connaissances propositionnelle est équivalent à **(R∘1)**-**(R∘6)**.

L'opération de révision selon la définition 6 (opération de révision), fondée sur le pré-ordre total "$\leq$" sur des modèles d'arbre, fournit directement la satisfaction du principe de changement minimal. En effet, $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$ ne retient que les modèles d'arbre dans $\mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ qui sont les plus proches des modèles d'arbre dans $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ selon la distance "$\triangle$" donnant le pré-ordre total "$\leq$". Cette observation nous permet d'obtenir alors le résultat disant que les postulats de révision implique un pré-ordre total "$\leq$" Il nous reste alors à prouver que l'opération de révision dans la définition 6 (opération de révision) satisfait les postulats de révision. Pour ce faire, nous utilisons les postulats de révision qui sont reformulés par Qi, Liu et Bell [Qi and Du, 2009] pour des ontologies en LD avec des

ensembles de modèles d'arbre $\mathsf{Mod}(\mathcal{O})$ et $\mathsf{Mod}(\mathcal{O}')$:

(Q1) $\mathsf{Mod}(\mathcal{O} \circ \mathcal{O}') \subseteq \mathsf{Mod}(\alpha)$ pour tous $\alpha \in \mathcal{O}'$;

(Q2) Si $\mathsf{Mod}(\mathcal{O}) \cap \mathsf{Mod}(\mathcal{O}') \neq \emptyset$, alors $\mathsf{Mod}(\mathcal{O} \circ \mathcal{O}') = \mathsf{Mod}(\mathcal{O}) \cap \mathsf{Mod}(\mathcal{O}')$;

(Q3) Si $\mathcal{O}'$ est cohérente alors $\mathsf{Mod}(\mathcal{O} \circ \mathcal{O}') \neq \emptyset$;

(Q4) Si $\mathsf{Mod}(\mathcal{O}_1) = \mathsf{Mod}(\mathcal{O}_2)$ et $\mathsf{Mod}(\mathcal{O}'_1) = \mathsf{Mod}(\mathcal{O}'_2)$, alors $\mathsf{Mod}(\mathcal{O}_1 \circ \mathcal{O}'_1) = \mathsf{Mod}(\mathcal{O}_2 \circ \mathcal{O}'_2)$;

(Q5) $\mathsf{Mod}(\mathcal{O} \circ \mathcal{O}') \cap \mathsf{Mod}(\mathcal{O}'') \subseteq \mathsf{Mod}(\mathcal{O} \circ \mathcal{O}' \cup \mathcal{O}'')$;

(Q6) Si $\mathsf{Mod}(\mathcal{O} \circ \mathcal{O}') \cap \mathsf{Mod}(\mathcal{O}'') \neq \emptyset$ alors $\mathsf{Mod}(\mathcal{O} \circ \mathcal{O}' \cup \mathcal{O}'') \subseteq \mathsf{Mod}(\mathcal{O} \circ \mathcal{O}') \cap \mathsf{MT}(\mathcal{O}'')$.

où l'ontologie résultante de la révision de $\mathcal{O}$ par $\mathcal{O}'$ est désignée par $\mathcal{O} \circ \mathcal{O}'$. Selon le corollaire 1 et la définition 6 (opération de révision), nous pouvons remplacer $\mathsf{Mod}(\mathcal{O})$ et $\mathsf{Mod}(\mathcal{O} \circ \mathcal{O}')$ avec $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ et $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$, respectivement, pour obtenir les postulats de révision qui se réfèrent uniquement aux structures calculables. En effet, les postulats **(Q1)**-**(Q6)** de Qi, Liu et Bell peuvent être reformulés dans notre contexte comme suit :

(D1) $\mathcal{I}(T) \models \alpha$ pour chaque $T \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')$ et chaque axiome $\alpha \in \mathcal{O}'$;

(D2) Si $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \neq \emptyset$,
alors $\mathsf{MT}(\mathcal{O}, \mathcal{O}') = \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$;

(D3) Si $\mathcal{O}'$ est cohérente alors $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \neq \emptyset$;

(D4) Si $\mathsf{MT}(\mathcal{O}_1, \mathsf{sub}(\mathcal{O}'_1)) = \mathsf{MT}(\mathcal{O}_2, \mathsf{sub}(\mathcal{O}'_2))$ et
$\mathsf{MT}(\mathcal{O}'_1, \mathsf{sub}(\mathcal{O}_1)) = \mathsf{MT}(\mathcal{O}'_2, \mathsf{sub}(\mathcal{O}_2))$, alors $\mathsf{MT}(\mathcal{O}_1, \mathcal{O}'_1) = \mathsf{MT}(\mathcal{O}_2, \mathcal{O}'_2)$;

(D5) $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \cap \mathsf{MT}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')) \subseteq \mathsf{MT}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'')$;

(D6) Si $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \cap \mathsf{MT}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')) \neq \emptyset$,
alors $\mathsf{MT}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'') \subseteq \mathsf{MT}(\mathcal{O}, \mathcal{O}') \cap \mathsf{MT}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}'))$.

Intuitivement, **(D1)** garantit que tous les axiomes de la nouvelle ontologie $\mathcal{O}'$ peuvent être déduits à partir du résultat de révision. **(D2)** énonce que l'ontologie initiale $\mathcal{O}$ n'est pas changée s'il n'y a aucun conflit. **(D3)** est une condition empêchant une révision de l'incohérence injustifiée. **(D4)** spécifie que l'opération de révision doit être indépendante de la syntaxe des ontologies. Le principe de changement minimal est assuré par **(D5)** et

(D6) car ils permettent de définir un pré-ordre total sur des modèles d'arbre [Katsuno and Mendelzon, 1991]. Nous pouvons prouver que tous les postulats sont toujours respectés dans notre cadre de travail.

Théorème 2. *L'opération de révision* $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$ *décrite dans la définition 6 satisfait les postulats* ***(D1)-(D6)****.*

Résumé de démonstration. **(D1)** : il peut être prouvé à partir de la définition 6 qui dit que $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$.

(D2) : Par la définition 6, si $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \neq \emptyset$, $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$ conserve uniquement les modèles d'arbre qui appartiennent à l'intersection de $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ et $\mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ car $T' \triangle T' = 0$ pour chaque T' appartenant à cette intersection.

(D3) : Par la définition 6, $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$ n'est jamais vide si $\mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ n'est pas vide.

(D4) : c'est une conséquence directe de la définition 6.

(D5) : Let $T' \in \mathsf{MT}(\mathcal{O}, \mathcal{O}') \cap \mathsf{MT}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}'))$. Par la définition 6, $T' \in \mathsf{MT}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}') \cup \mathsf{sub}(\mathcal{O}'')) \subseteq \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}') \cup \mathsf{sub}(\mathcal{O}''))$, et il existe un $T \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ qui est le plus proche de T' car $T' \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')$.

(D6) : soit $T \in \mathsf{MT}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'')$. Par la définition 6, on a $T \in \mathsf{MT}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}') \cup \mathsf{sub}(\mathcal{O}'))$. Pour montrer que $T \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')$, on utilise $T_0 \in \mathsf{MT}(\mathcal{O}, \mathcal{O}') \cap \mathsf{MT}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}'))$ et le pré-ordre total sur les modèles d'arbre.

Une preuve complète peut être trouvée en annexe. □

3.4 Calcul de l'ontologie de révision

Dans cette section, nous présentons une procédure pour la construction d'une ontologie $\mathcal{O}^*$ en $\mathcal{SHIQ}$ qui admet au moins les modèles d'arbre dans $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$.

3.4.1 Approximation supérieure

Il s'est avéré [De Giacomo *et al.*, 2007] qu'il peut ne pas exister une ontologie en DL-Lite qui admette exactement un ensemble donné de modèles. Par l'exemple 10, nous montrons que c'est également le cas pour des ontologies en $\mathcal{SHIQ}$.

Exemple 10. Reconsidérons l'exemple 9 avec $\mathsf{MT}(\mathcal{O}, \mathcal{O}') = \{T_1', T_2'\}$. Supposons qu'il existe $\widehat{\mathcal{O}}$ avec $\mathsf{sub}(\widehat{\mathcal{O}}) = \{\mathsf{User}, \neg\mathsf{User}, \mathsf{BakerVideo}, \neg\mathsf{BakerVideo}, \mathsf{PastryVideo}, \neg\mathsf{PastryVideo},$

$\exists\mathsf{hasSeen}.(\mathsf{BakerVideo} \sqcup \mathsf{PastryVideo}), \forall\mathsf{hasSeen}.(\neg\mathsf{BakerVideo} \sqcap \neg\mathsf{PastryVideo})\}$ qui admet les deux T_1' et T_3' comme des modèles d'arbre. En appliquant le nouvel algorithme de tableau à $\widehat{\mathcal{O}}$, $\mathsf{MT}(\widehat{\mathcal{O}})$ doit contenir T_1', T_3' et un autre T_6' ayant un nœud $\{U_1'\}$ avec $L(U_1') = \{\neg\mathsf{User}, \mathsf{BakerVideo}, \mathsf{PastryVideo}, \forall\mathsf{hasSeen}.(\neg\mathsf{BakerVideo} \sqcap \neg\mathsf{PastryVideo})\}$, ce qui est contradictoire.

Pour résoudre ce problème, nous empruntons la notion d'*approximation maximale* proposée par De Giacomo *et al.* [De Giacomo *et al.*, 2007]. Cette notion peut être reformulée dans notre cadre de recherche comme suit :

Définition 7 (**Approximation maximale**). *Soient $\mathcal{O}$ et $\mathcal{O}'$, deux ontologies cohérentes en $\mathcal{SHIQ}$ avec l'opération de révision $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$. Nous utilisons $\mathcal{S}(\mathcal{O}'')$ pour désigner la signature d'une ontologie $\mathcal{O}''$. Une ontologie $\mathcal{O}^*$ est une* approximation maximale *de $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$ si*

1. $\mathcal{S}(\mathcal{O}^*) \subseteq \mathcal{S}(\mathcal{O}) \cup \mathcal{S}(\mathcal{O}')$,
2. $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}^*)$,
3. *Il n'existe aucune ontologie $\mathcal{O}''$ telle que $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}'') \subset \mathsf{MT}(\mathcal{O}^*)$.* ◁

La définition 7 (approximation maximale) fournit une meilleure approximation des ontologies en $\mathcal{SHIQ}$ que l'on doit construire de telle façon qu'elle admet tous les modèles d'arbre dans $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$. Un point intéressant est que si une telle approximation maximale existe alors l'équivalence sémantique est unique en son genre. En effet, supposons qu'il existe une approximation maximale $\mathcal{O}''$ telle que $\mathsf{MT}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}^*)) \neq \mathsf{MT}(\mathcal{O}^*, \mathsf{sub}(\mathcal{O}''))$. Nous pouvons montrer que $\mathsf{sub}(\mathcal{O}^*) = \mathsf{sub}(\mathcal{O}'') = \mathsf{sub}(\mathcal{O} \cup \mathcal{O}')$. Puis, par le corollaire 1, $\mathsf{MT}(\mathcal{O}'' \cup \mathcal{O}^*) = \mathsf{MT}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}^*)) \cap \mathsf{MT}(\mathcal{O}^*, \mathsf{sub}(\mathcal{O}'')) = \mathsf{MT}(\mathcal{O}'') \cap \mathsf{MT}(\mathcal{O}^*)$

(car $\mathsf{sub}(\mathcal{O}^*) = \mathsf{sub}(\mathcal{O}'')$), nous avons donc $\mathsf{MT}(\mathcal{O}'' \cup \mathcal{O}^*) \subset \mathsf{MT}(\mathcal{O}^*)$ ou $\mathsf{MT}(\mathcal{O}'' \cup \mathcal{O}^*) \subset \mathsf{MT}(\mathcal{O}'')$, ce qui contredit la condition 3 de la définition 7 (approximation maximale). De plus, l'existence de $\mathcal{O}''$ de la condition 3 implique la condition 1, *i.e.*, $\mathcal{S}(\mathcal{O}'') \subseteq \mathcal{S}(\mathcal{O}) \cup \mathcal{S}(\mathcal{O}')$. Par la suite, nous montrerons qu'une telle approximation maximale existe en réalité et proposerons une procédure pour la construire.

3.4.2 Construction de l'ontologie de révision

Définition 8 (Ontologie de révision). *Soient $\mathcal{O} = (\mathcal{T}, \mathcal{R})$ et $\mathcal{O}' = (\mathcal{T}', \mathcal{R}')$, deux ontologies cohérentes en $\mathcal{SHIQ}$ avec $\mathsf{MT}(\mathcal{O}, \mathcal{O}') = \{T_1, \cdots, T_n\}$ où $T_i = \langle V_i, L_i, E_i, \widehat{x}_i \rangle$ pour $1 \leq i \leq n$. Une ontologie $\mathcal{O}^* = (\widehat{\mathcal{T}}, \widehat{\mathcal{R}})$ de la révision de $\mathcal{O}$ par $\mathcal{O}'$ est définie comme suit :*

— $\widehat{\mathcal{R}} := \mathcal{R}'$

— $\widehat{\mathcal{T}} := \mathcal{T}' \cup \{\top \sqsubseteq \bigsqcup_{\langle V_i, L_i, E_i, \widehat{x}_i \rangle \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')} (\bigsqcup_{x \in V_i} (\bigsqcap_{C \in L_i(x)} C))\}$ ◁

La seule différence entre $\mathcal{O}'$ et $\mathcal{O}^*$, c'est le nouvel axiome construit de $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$. Cet axiome permet de sélectionner à partir de $\mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ des modèles d'arbre qui sont les plus proches de $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$. Le "père" de l'axiome contient tous les concepts se trouvant dans chaque nœud de chacun des modèles d'arbre $T_i = \langle V_i, L_i, E_i, \widehat{x_i} \rangle \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')$.

Exemple 11. En continuant l'exemple 9, nous construisons à partir de $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$ une ontologie $\mathcal{O}^*$ qui admet deux modèles d'arbre T_1' et T_3' selon la définition d'ontologie de révision. Donc, $\mathcal{O}^*$ contient les axiomes suivants :

$$\top \sqsubseteq (\mathsf{BakerVideo} \sqcap \mathsf{PastryVideo}) \sqcup \mathsf{User} \tag{3.6}$$

(qui sont conservés de $\mathcal{O}'$), et un nouvel axiome :

$\top \sqsubseteq (\mathsf{User} \sqcap \neg\mathsf{BakerVideo} \sqcap \neg\mathsf{PastryVideo} \sqcap \exists \mathsf{hasSeen}.(\mathsf{BakerVideo} \sqcup \mathsf{PastryVideo})) \sqcup$

$(\mathsf{BakerVideo} \sqcap \mathsf{PastryVideo} \sqcap \neg\mathsf{User} \sqcap \forall\mathsf{hasSeen}.(\neg\mathsf{BakerVideo} \sqcap \neg\mathsf{PastryVideo})) \sqcup$

$(\mathsf{User} \sqcap \neg\mathsf{BakerVideo} \sqcap \mathsf{PastryVideo} \sqcap \exists\mathsf{hasSeen}.(\mathsf{BakerVideo} \sqcup \mathsf{PastryVideo}))$.

Nous formulons et montrons le résultat le plus important affirmant que l'ontologie de révision $\mathcal{O}^*$ dans la définition 8 (ontologie de révision) satisfait les conditions d'une approximation maximale. Notre argument repose fortement sur le comportement spécifique de la sat-règle. En effet, cette règle permet d'obtenir un modèle d'arbre construit par l'algorithme de tableau sans indiquer une séquence précise d'application de règles d'expansion.

Théorème 3. *Soient $\mathcal{O}$ et $\mathcal{O}'$, deux ontologies cohérentes en $\mathcal{SHIQ}$. L'ontologie de révision $\mathcal{O}^*$ de $\mathcal{O}$ par $\mathcal{O}'$ est une approximation maximale de $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$. En outre, la taille de $\mathcal{O}^*$ est bornée par une fonction doublement exponentielle de la taille de $\mathcal{O}$ et $\mathcal{O}'$.*

Résumé de démonstration. Par la construction, $\mathcal{S}(\mathcal{O}^*) \subseteq \mathcal{S}(\mathcal{O}) \cup \mathcal{S}(\mathcal{O}')$ et $\mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}') = \mathsf{sub}(\mathcal{O}^*)$. Soit $T \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')$. Comme $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$, il existe une séquence Seq_T d'applications de règles exécutées par le nouvel algorithme de tableau sur $\mathcal{O}'$ avec $\mathsf{sub}(\mathcal{O})$ telles que Seq_T permette de construire T. Nous utilisons Seq_T pour guider la construction d'une séquence $Seq_{T'}$ d'applications de règles exécutées par le nouvel algorithme de tableau sur $\mathcal{O}^*$ telles que $Seq_{T'}$ permette de construire un modèle d'arbre $T' = T$. Donc, $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}^*)$. Pour prouver la condition 3, nous montrons que s'il existe une ontologie $\mathcal{O}''$ avec $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}'') \subseteq \mathsf{MT}(\mathcal{O}^*)$ alors $\mathsf{MT}(\mathcal{O}'') = \mathsf{MT}(\mathcal{O}^*)$. D'abord, montrons que $\mathsf{sub}(\mathcal{O}'') = \mathsf{sub}(\mathcal{O}^*)$. Le reste peut être fait en utilisant le même argument sur des séquences d'applications de règles exécutées par le nouvel algorithme de tableau sur $\mathcal{O}''$ et $\mathcal{O}^*$. En effet, si l'on connait une séquence Seq_T avec le comportement spécifique de la sat-règle (afin de construire un modèle d'arbre T pour $\mathcal{O}''$) et $\mathsf{sub}(\mathcal{O}'') = \mathsf{sub}(\mathcal{O}^*)$, il est possible de reproduire Seq_T pour construire le même modèle d'arbre $T' = T$ pour $\mathcal{O}^*$, et inversement. De plus, il existe au plus un nombre doublement exponentiel de modèles d'arbre $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$, et la taille de chaque modèle d'arbre est bornée par une fonction doublement exponentielle. Une preuve complète peut être trouvée en annexe. □

En se basant sur la définition 8 (ontologie de révision), nous pouvons définir une procédure

Algorithme 2 : Algorithme de calcul de l'ontologie de révision $\mathcal{O}^*$

Entrée : $\mathcal{O} = (\mathcal{T}, \mathcal{R})$, $\mathcal{O}' = (\mathcal{T}', \mathcal{R}')$: deux ontologies cohérentes en $\mathcal{SHIQ}$
Sortie : $\mathcal{O}^* = (\widehat{\mathcal{T}}, \widehat{\mathcal{R}})$: l'ontologie de révision de $\mathcal{O}$ par $\mathcal{O}'$

1 Appliquer le nouvel algorithme de tableau à $\mathcal{O}$ et $\mathcal{O}'$ pour construire $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ et $\mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$;
2 Calculer $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$ à partir de $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ et $\mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ selon Définition 6;
3 **pour chaque** $\alpha \in \mathcal{R}'$ **faire**
4 Ajouter α dans $\widehat{\mathcal{R}}$;
5 **pour chaque** $\alpha \in \mathcal{T}'$ **faire**
6 Ajouter α dans $\widehat{\mathcal{T}}$;
7 Ajouter dans $\widehat{\mathcal{T}}$ l'axiome de concept $\top \sqsubseteq \bigsqcup_{\langle V_i, L_i, E_i, \widehat{x}_i \rangle \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')} \left(\bigsqcup_{x \in V_i} \left(\bigsqcap_{C \in L_i(x)} C \right) \right)$;
8 **retourner** $\mathcal{O}^*$;

(*cf.* Algorithme 2) pour calculer l'ontologie de révision $\mathcal{O}^*$ de $\mathcal{O}$ par $\mathcal{O}'$. La construction de l'axiome par la ligne 7 est effectuée en parcourant chaque arbre $T_i \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')$ pour retirer les étiquettes de nœud de T_i. Le subsumeur de cet axiome est une disjonction dont chaque disjoint correspond à l'étiquette de chaque nœud $x \in V_i$ avec $T_i = (V_I, L_i, E_i, \widehat{x}_i)$.

3.5 Conclusion

Nous avons présenté dans ce chapitre une nouvelle approche pour la révision d'ontologie en $\mathcal{SHIQ}$. Cette opération de révision garantit le changement minimal et l'ontologie résultante reste exprimable dans la logique de l'ontologie initiale. Une caractéristique intéressante de notre approche est d'introduire des structures finies, à savoir des *arbres de complétion*, pour caractériser un ensemble de modèles d'une ontologie en $\mathcal{SHIQ}$. Ces structures sont construites par un algorithme de tableau avec une nouvelle règle d'expansion, à savoir la sat-règle. Bien que le comportement non-déterministe de cette règle trahit certaines "bonnes" caractéristiques d'algorithmes de tableau classiques, cet inconvénient est justifié par l'obtention d'une opération de révision qui garantit tous les postulats de révision (par exemple, le principe de changement minimal), et d'une ontologie de révision exprimable dans la logique des ontologies initiales.

La principale limitation de notre approche présentée dans ce chapitre est d'omettre les individus représentés dans ces ontologies. Cependant, cette limitation sera outrepassée dans le prochain chapitre.

Chapitre 4

Révision d'ontologies d'expressivité $\mathcal{SHIQ}$ avec individus

Dans ce chapitre, nous proposons une extension de l'approche fondée sur les modèles présentée dans le chapitre précédent pour réviser des ontologies en $\mathcal{SHIQ}$ avec individus. La construction de notre procédure de révision est fondée sur les points suivants : (i) utiliser des graphes de complétion générés par un nouvel algorithme pour caractériser la sémantique d'une ontologie en $\mathcal{SHIQ}$. Cet algorithme doit construire un ensemble de graphes de complétion, noté $\mathsf{FM}(\mathcal{O})$, pour une ontologie $\mathcal{O}$ en considérant tous les cas non-déterministes intrinsèques au lieu de construire un seul graphe de complétion comme ce que les algorithmes de tableau existants font ; (ii) définir une distance sur un ensemble de graphes de complétion pour aborder le principe du changement minimal. Étant donné une ontologie $\mathcal{O}'$ contenant de nouveaux axiomes devant être pris en compte lors de la révision, cette distance peut aider à choisir des graphes de complétion à partir de $\mathsf{FM}(\mathcal{O}')$ de telle façon qu'ils soient sémantiquement les plus proches de ceux de $\mathsf{FM}(\mathcal{O})$. Une ontologie de révision de $\mathcal{O}$ par $\mathcal{O}'$ doit admettre les graphes de complétion choisis comme modèles ; (iii) introduire la notion d'approximation d'ontologie pour surmonter le problème de l'inexprimabilité. Notre procédure de révision renvoie une approximation d'ontologie exprimable en $\mathcal{SHIQ}$ et la plus petite en terme de sémantique.

Pour illustrer l'idée sous-jacente à cette construction, nous considérons l'exemple 12.

Exemple 12. Étant donnée une ontologie UNI (*cf.* Tableau 4.1).

α_1 : Professor $\sqsubseteq$ Researcher $\sqcup$ Expert	Un professeur est un chercheur ou un expert
α_2 : Professor $\sqsubseteq \exists$supervises.Student	Un professeur supervise au moins un étudiant
α_3 : Professor $\sqsubseteq$ ($\geq$ 2 teaches.Course)	Un professeur enseigne au moins deux cours
β : Professor(Alex)	Alex est un professeur

TABLEAU 4.1 – Ontologie UNI

Supposons que les chercheurs et les experts ne supervisent aucun étudiant, nous devons alors ajouter dans UNI cette connaissance formulée par les axiomes suivants :

$$(\delta_1) : \text{Researcher} \sqsubseteq \forall \text{supervises}.(\neg \text{Student})$$
$$(\delta_2) : \text{Expert} \sqsubseteq \forall \text{supervises}.(\neg \text{Student})$$

Cependant, la présence de δ_1 et δ_2 rendra UNI incohérente, et cela nécessite une révision pour maintenir la cohérence d'UNI. Une des façons d'appliquer la révision est de supprimer certains axiomes d'UNI. Intuitivement, on peut éliminer soit α_1 soit α_2 pour maintenir sa cohérence. Dans le cas où α_1 est retiré alors l'ontologie obtenue $\widehat{\mathcal{O}} = \{\alpha_2, \alpha_3, \beta, \delta_1, \delta_2\}$ est cohérente. Cependant, la connaissance "Un professeur est un expert" dans α_1 ne contredit pas l'ontologie $\widehat{\mathcal{O}}$ mais elle a été enlevée en même temps qu'α_1. En d'autres termes, l'objectif est de construire une nouvelle ontologie $\mathcal{O}^*$ "compatible" avec les axiomes d'UNI telle que $\mathcal{O}^*$ est sémantiquement la plus proche possible d'UNI ; ce qui signifie un changement minimal. Nous pouvons vérifier que les forêts de complétion $\mathcal{F}_1$, $\mathcal{F}_2$ de la figure 4.1 donnent des modèles d'UNI. Dans ce cas, chaque nœud étiqueté représente un individu et chaque arc représente une relation entre deux individus.

De même, nous pouvons vérifier que les deux forêts de la figure 4.2 donnent des modèles de $\{\delta_1, \delta_2\}$.

Si nous définissons une distance entre les forêts de complétion en fonction de la similarité structurelle, il serait plausible de dire que $\mathcal{F}_1'$ est plus proche de $\mathcal{F}_1$ et $\mathcal{F}_2$ que $\mathcal{F}_2'$. Donc, une ontologie de révision $\mathcal{O}^*$ devrait admettre $\mathcal{F}_1'$ plutôt que $\mathcal{F}_2'$.

Pour étendre ce que nous avons défini dans le chapitre précédent, nous redéfinissons la notation de sous-concepts pour des ontologies en $\mathcal{SHIQ}$ avec individus selon la définition 9.

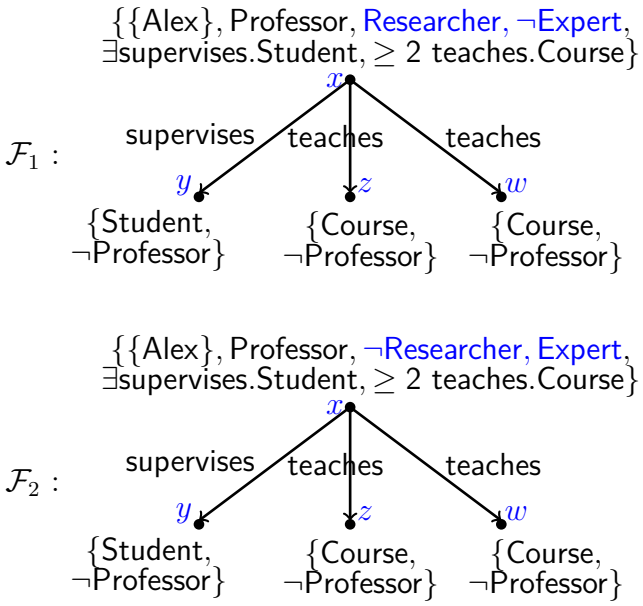


FIGURE 4.1 – Forêts de complétion donnant des modèles d'UNI

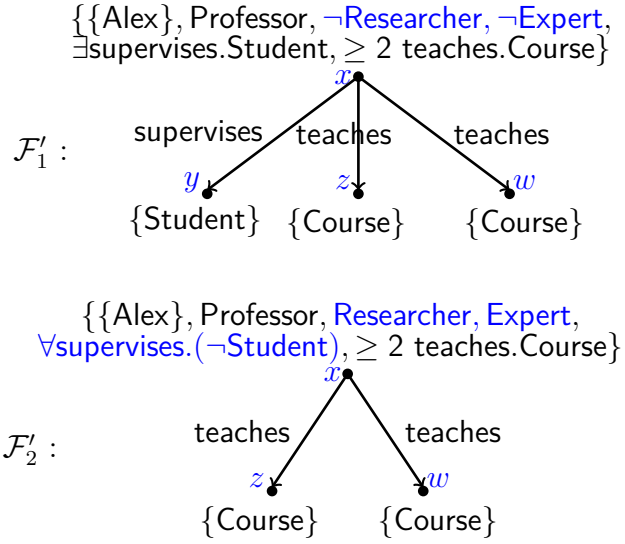


FIGURE 4.2 – Forêts de complétion donnant des modèles de $\{\delta_1, \delta_2\}$

Définition 9 (Sous-concepts). *Soit* $\mathcal{O} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ *une ontologie en* $\mathcal{SHIQ}$ *avec* $\mathcal{S}(\mathcal{O}) = \mathbf{R} \cup \mathbf{C} \cup \mathbf{I}$, *un ensemble* $\mathsf{sub}(\mathcal{O})$ *est défini de manière inductive comme suit :*

$$\mathsf{sub}(\mathcal{O}) = \mathsf{sub}(\mathcal{T}) \cup \mathsf{sub}(\mathcal{A}) \cup \{\dot{\neg} C \mid C \in \mathsf{sub}(\mathcal{T}) \cup \mathsf{sub}(\mathcal{A})\}$$

$$\mathsf{sub}(\mathcal{T}) = \bigcup_{C \sqsubseteq D \in \mathcal{T}} \mathsf{sub}(\mathsf{nnf}(\neg C \sqcup D))$$

$$\mathsf{sub}(\mathcal{A}) = \{\mathsf{sub}(\mathsf{nnf}(C)) \mid C(a) \in \mathcal{A}\}$$

$$\mathsf{sub}(C) = \begin{cases} \{C, \neg C\} & si\ C \in \mathbf{C} \\ \mathsf{sub}(E) \cup \mathsf{sub}(F) & si\ C \in \{E \sqcap F, E \sqcup F\} \\ \{C\} \cup \{\exists R'.E \mid R \sqsubseteq R'\} \cup \mathsf{sub}(E) & si\ C = \exists R.E \\ \{C\} \cup \{\forall R'.E \mid R \sqsubseteq R'\} \cup \mathsf{sub}(E) & si\ C = \forall R.E \\ \{C\} \cup \{\geq nR'.E \mid R \sqsubseteq R'\} \cup \mathsf{sub}(E) & si\ C = (\geq nR.E) \\ \{C\} \cup \mathsf{sub}(E) & si\ C = (\leq nR.E) \end{cases}$$

◁

4.1 Sémantique d'ontologies par forêts de complétion

Une approche fondée sur les modèles [Alchourrón *et al.*, 1985] pour la révision d'une ontologie devrait modifier les contraintes sémantiques de l'ontologie en manipulant un ensemble de ses modèles. Cependant, l'ensemble de tous les modèles d'une ontologie peut être infini et il existe une ontologie en $\mathcal{SHIQ}$ qui n'admet que des modèles infinis. Ceci nous amène à proposer une méthode qui permet de caractériser la sémantique d'une ontologie en $\mathcal{SHIQ}$ en utilisant un ensemble fini de *graphes de complétion*. Horrocks et ses collèges [Horrocks *et al.*, 2000] ont proposé un algorithme de tableau pour la vérification de la cohérence d'une ontologie $\mathcal{O}$ en $\mathcal{SHIQ}$. Cet algorithme tente de construire un graphe étiqueté fini, à savoir une *forêt de complétion*, à partir duquel on peut inventer un modèle pour $\mathcal{O}$. L'algorithme retourne "OUI" si une telle forêt de complétion est construit, et retourne "NON" s'il ne construit pas une telle forêt de complétion après avoir pris en considération tous les cas possibles de non-déterminismes. Pour pouvoir caractériser la sémantique d'une ontologie, nous avons besoin plutôt d'un ensemble de forêts de complétion qui décrivent différents modèles résultant de constructeurs logiques non déterministes qu'une seule forêt de complétion. Pour cela, nous adaptons l'algorithme tableau d'Horrocks et ses collègues [Horrocks *et al.*, 2000] de manière à explorer tous les cas intrinsèques non-déterministes.

Définition 10 (Forêt de complétion). *Soit* $\mathcal{O} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ *une ontologie en* $\mathcal{SHIQ}$, *une* forêt de complétion $\mathcal{F}$ *pour* $\mathcal{O}$ *est un tuple* $\mathcal{F} = (G, T\langle\hat{x}_1\rangle, \ldots, T\langle\hat{x}_n\rangle)$ *où*

— $G = (\mathbf{V}, \mathbf{E}, \mathbf{L})$ *est un graphe orienté avec* $\mathbf{V}$ *un ensemble de nœuds,* $\mathbf{E}$ *un ensemble d'arcs et* $\mathbf{L}$ *une fonction d'étiquetage qui associe à chaque nœud* $x \in \mathbf{V}$ *un ensemble* $\mathbf{L}(x) \subseteq \mathsf{sub}(\mathcal{O})$ *et à chaque arc* $\langle \widehat{x}, \widehat{y} \rangle \in \mathbf{E}$ *un ensemble* $\mathbf{L}(\langle \widehat{x}, \widehat{y} \rangle) \subseteq \mathbf{R} \cup \mathbf{R}_{\mathsf{I}}$. *Un nœud* $\widehat{y} \in \mathbf{V}$ *s'appelle un* R*-voisin de* $\widehat{x} \in \mathbf{V}$ *si* $R \in \mathbf{L}(\langle \widehat{x}, \widehat{y} \rangle)$ *ou* $\mathsf{Inv}(R) \in \mathbf{L}(\langle \widehat{y}, \widehat{x} \rangle)$. *Par abus de notation, nous utilisons* $L(\widehat{x})$ *et* $L(\langle \widehat{x}, \widehat{y} \rangle)$ *au lieu de* $\mathbf{L}(\widehat{x})$ *et* $\mathbf{L}(\langle \widehat{x}, \widehat{y} \rangle)$.

— *Chaque* $T\langle \widehat{x}_i \rangle = (V_i, E_i, L_i)$ $(1 \leq i \leq n)$ *est un arbre "enraciné" par* $\widehat{x}_i$ *appartenant à* G *(i.e.* $\widehat{x}_i \in \mathbf{V}$*),* V_i *un ensemble de nœuds,* E_i *un ensemble d'arcs, et* L_i *une fonction d'étiquetage qui associe à chaque nœud* $x \in V_i$ *un ensemble* $L_i(x) \subseteq \mathsf{sub}(\mathcal{O})$ *et à chaque arc* $\langle x, y \rangle \in E_i$ *un ensemble* $L_i(\langle x, y \rangle) \subseteq \mathbf{R} \cup \mathbf{R}_{\mathsf{I}}$. *Par abus de notation, nous utilisons* $L(x)$ *et* $L(\langle x, y \rangle)$ *au lieu de* $L_i(x)$ *et* $L_i(\langle x, y \rangle)$.

Si deux nœuds $x, y \in V_i$ *(d'un certain arbre* $T\langle \widehat{x}_i \rangle = (V_i, E_i, L_i)$*) reliés par un arc* $\langle x, y \rangle \in E_i$*, alors* y *s'appelle un* successeur *de* x*, et* x *s'appelle un* prédécesseur *de* y *; et* ancêtre *est la fermeture transitive de* prédécesseur. *Un nœud* y *s'appelle un* R*-successeur de* x *si, pour un certain rôle* R' *avec* $R' \sqsubseteq R$, $R' \in L(\langle x, y \rangle)$ *; et* x *s'appelle un* R*-prédécesseur de* y *si* y *est un* R*-successeur de* x. *Un nœud* y *s'appelle un* R*-voisin de* x *si* y *est un* R*-successeur ou* x *est un* $\mathsf{Inv}(R)$*-successeur de* y.

Un nœud $x \in V_i$ *est* bloqué *par un nœud* $y \in V_i$ *s'il n'est pas un nœud racine et a des ancêtres* x', y *et* y' *tels que (i)* y *n'est pas un nœud racine, (ii)* x *est un successeur de* x' *et* y *est un successeur de* y'*, (iii)* $L(x) = L(y)$*,* $L(x') = L(y')$*, et (iv)* $L(\langle x', x \rangle) = L(\langle y', y \rangle)$.

En outre, il existe une relation d'inégalité $\neq$ *et une relation d'égalité* $\doteq$ *définies sur les nœuds dans* $\mathcal{F}$. *Pour un rôle* S*, un concept* C *et un nœud* x *dans* $\mathcal{F}$*, nous définissons l'ensemble* $S^{\mathcal{F}}(x, C)$ *de* S*-voisins de* x *comme suit :* $S^{\mathcal{F}}(x, C) = \{y \mid y$ *est un* S*-voisin de* x *et* $X \subseteq L(y)$ *pour un certain* $X \in \mathsf{Flat}(C)\}$.

$\mathcal{F}$ *contient un* clash *si (i) il y a certains nœuds* x *dans* $\mathcal{F}$ *tels que soit* $\{A, \neg A\} \subseteq L(x)$ *pour certains noms de concept* $A \in \mathbf{C}$*, ou (ii)* $(\leq nS.C) \in L(x)$ *et il y a* $(n+1)$ S*-voisins* $y_1, \cdots, y_{n+1}$ *de* x *avec* $y_i \neq y_j$ *et* $X \subseteq L(y_i)$ *pour certains* $X \in \mathsf{Flat}(C)$ *et tout* $1 \leq i < j \leq (n+1)$.

$\exists$-règle : si 1. $\exists S.C \in L(x)$, x n'est pas bloqué, et
2. x n'a aucun S-voisin y t.q. $X \subseteq L(y)$ pour certains $X \in \mathsf{Flat}(C)$
alors créer un nouveau nœud y avec $L(\langle x, y\rangle) \leftarrow \{S\}$ et $L(y) \leftarrow X$ pour $X \in \mathsf{Flat}(C)$.

$\forall$-règle : si 1. $\forall S.C \in L(x)$, et
2. il y un S-voisin y de x t.q. $X \not\subseteq L(y)$ pour tout $X \in \mathsf{Flat}(C)$
alors $L(y) \leftarrow L(y) \cup X$ pour certains $X \in \mathsf{Flat}(C)$.

$\forall_+$-règle : si 1. $\forall S.C \in L(x)$,
2. il y a un R avec $\mathsf{Trans}(R)$ t.q. $R \sqsubseteq S$, et
3. il y a un R-voisin y de x t.q. $\forall R.C \notin L(y)$
alors $L(y) \leftarrow L(y) \cup \{\forall R.C\}$.

$\geq$-règle : si 1. $(\geq nS.C) \in L(x)$, x n'est pas bloqué, et
2. x n'a pas n S-voisins $y_1, \cdots, y_n$ t.q. $X \subseteq L(y_i)$ pour certains
$X \in \mathsf{Flat}(C)$ et $y_i \neq y_j$ pour $0 \leq i < j \leq n$
alors créer n nouveaux nœuds $y_1, \cdots, y_n$ avec $L(\langle x, y_i\rangle) \leftarrow \{S\}$,
$L(y_i) \leftarrow X$ pour certain $X \in \mathsf{Flat}(C)$ et $y_i \neq y_j$ pour $1 \leq i < j \leq n$.

$\leq$-règle : si 1. $(\leq nS.C) \in L(x)$,
2. x a $n+1$ S-voisins $y_0, \ldots, y_n$ t.q. $X \subseteq L(y_i)$ pour certains $X \in \mathsf{Flat}(C)$,
3. il y a deux S-voisins y, z de x avec $X_1 \subseteq L(y)$, $X_2 \subseteq L(z)$
pour certains $X_1, X_2 \in \mathsf{Flat}(C)$, y n'est pas un ancêtre de z, et pas de $y \neq z$
alors (i) $L(z) \leftarrow L(z) \cup L(y)$ et $L(\langle x, y\rangle) \leftarrow \emptyset$
(ii) si z est un ancêtre de x
alors $L(\langle z, x\rangle) \leftarrow L(\langle z, x\rangle) \cup \{\mathsf{Inv}(R) \mid R \in L(\langle x, y\rangle)\}$;
sinon $L(\langle x, z\rangle) \leftarrow L(\langle x, z\rangle) \cup L(\langle x, y\rangle)$, et
(iii) ajouter $u \neq z$ pour tous u t.q. $u \neq y$.

$\leq_r$-règle : si 1. $(\leq nS.C) \in L(x)$,
2. x a $n+1$ S-voisins $y_0, \cdots, y_n$ t.q. $X \subseteq L(y_i)$ pour certains $X \in \mathsf{Flat}(C)$,
3. $y_i \neq y_j$ n'est pas satisfait pour $0 \leq i < j \leq n$ où y_i, y_j sont des nœuds racines,
alors (i) $L(y_i) \leftarrow L(y_i) \cup L(y_j)$,
(ii) pour toute arête $\langle y_j, w\rangle$,
si l'arête $\langle y_i, w\rangle$ n'existe pas, créer celle-ci avec $L(\langle y_i, w\rangle) = \emptyset$;
faire $L(\langle y_i, w\rangle) \leftarrow L(\langle y_i, w\rangle) \cup L(\langle y_j, w\rangle)$,
(iii) pour toute arête $\langle w, y_j\rangle$,
si l'arête $\langle w, y_i\rangle$ n'existe pas, créer celle-ci avec $L(\langle w, y_i\rangle) = \emptyset$;
faire $L(\langle w, y_i\rangle) \leftarrow L(\langle w, y_i\rangle) \cup L(\langle w, y_j\rangle)$,
(iv) faire $L(y_j) \leftarrow \emptyset$ et supprimer toute arête à/de y_j,
(v) faire $u \neq y_i$ pour tout u avec $u \neq y_j$,
(vi) faire $y_j \doteq y_i$.

---

sat-règle : si sat-règle n'est jamais appliquée à x

alors choisir un sous-ensemble $S \subseteq \mathsf{sub}(\mathcal{O})$ t.q. $L(x) \cup \bigcup_{X \in \mathsf{Flat}(\mathsf{nnf}(\neg C \sqcup D)), C \sqsubseteq D \in \mathcal{T}} X \subseteq S$,
et faire $L(x) \leftarrow S \cup \bar{S}$ où $\bar{S} = \{\dot{\neg} C \mid C \in \mathsf{sub}(\mathcal{O}) \setminus S\}$.

FIGURE 4.3 – Règles d'expansion pour $\mathcal{SHIQ}$

4.1.1 Nouvel algorithme de tableau avec individus

En se fondant sur le travail d'Horrocks, Sattler et Tobies [Horrocks *et al.*, 2000], nous concevons un nouvel algorithme de tableau afin de construire une forêt de complétion en utilisant les règles d'expansion de la figure 4.3. Il existe deux différences principales entre les règles de la figure 4.3 et celles présentées dans un algorithme de tableau standard : (i) l'absence de règles de conjonction et de disjonction. Selon le lemme 1, l'application de la fonction Flat à un nouveau concept ajouté à l'étiquette d'un nœud supprime toutes les conjonctions et disjonctions au plus haut niveau de ce concept ; (ii) la présence des sat-, $\mathsf{sat}_{\mathcal{R}}$-règles ($\mathsf{sat}$ désigne *saturate* en anglais). Le choix d'un sous-ensemble $S \subseteq \mathsf{sub}(\mathcal{O})$ dans la sat-règle doit inclure tous les concepts aplatis des axiomes GCI (tels que ceux ajoutés par la $\sqsubseteq$-règle [Horrocks *et al.*, 2000]). De plus, la sat-règle ajoute dans chaque étiquette de nœud soit C ou $\doteq\neg C$ pour chaque $C \in \mathsf{sub}(\mathcal{O})$. Comme conséquence directe de la définition de la sat-règle, ces comportements sont très non déterministes puisque le nombre de possibilités pour choisir un sous-ensemble à partir de sub est limité par une fonction exponentielle dans la cardinalité de sub. Cependant, ils sont nécessaires pour construire une ontologie d'approximation à partir d'un ensemble de forêts de complétion et un ensemble de sous-concepts 4.3).

Pour une ontologie d'entrée $\mathcal{O} = \{\mathcal{T}, \mathcal{R}, \mathcal{A}\}$, notre algorithme de tableau commence par initialiser une forêt de complétion $\mathcal{F}$ avec seulement des nœuds racines et des arêtes. Cette partie de $\mathcal{F}$ représente les individus et les assertions définies dans $\mathcal{A}$. L'algorithme applique des règles de la figure 4.3 à chaque nœud jusqu'à ce qu'aucune des règles ne soit applicable à chaque nœud. Dans ce cas, $\mathcal{F}$ s'appelle *complète*. Si $\mathcal{F}$ ne contient pas de clash, elle s'appelle *non-clash*. Plus précisément, $\mathcal{F}$ contient un nœud racine $\widehat{x}_i$ pour chaque individu $a_i \in \mathbf{I}$ se trouvant dans $\mathcal{A}$, et une arête entre deux nœuds racines $\langle \widehat{x}_i, \widehat{x}_j \rangle$ si $\mathcal{A}$ contient une assertion $R(a_i, a_j)$ pour un certain rôle R. L'étiquette de ces nœuds racines et les arêtes entre eux avec les relations $\not\doteq$ et $\doteq$ est initialisée comme suit : $L(\widehat{x}_i) := \{\{a_i\} \cup X_k \in X \mid C(a_i) \in \mathcal{A}, X \in \mathsf{Flat}(C)\}$, $L(\langle \widehat{x}_i, \widehat{x}_j \rangle) := \{R \mid R(a_i, a_j) \in \mathcal{A}\}$, $\widehat{x}_i \not\doteq \widehat{x}_j$ si et seulement si $a_i \not\doteq a_j \in \mathcal{A}$, et la $\doteq$-relation est initialisée comme vide. Les applications des règles de génération ($\exists$- et $\geq$-règles) peuvent créer de nouveaux nœuds dont l'étiquette est entièrement remplie

par la sat-règle,ou partiellement remplie par les $\forall$-, $\forall_+$-règles. Après avoir appliqué la sat-règle à un nouveau nœud, son étiquette n'est plus modifiée puisque cette règle sature son étiquette en ajoutant soit $X \in \mathsf{Flat}(C)$, soit $X \in \mathsf{Flat}(\dot{\neg}C)$ pour chaque concept $C \in \mathsf{sub}(\mathcal{O})$. Ceci explique pourquoi les $\sqsubseteq$- et ch-règles dans un algorithme de tableau standard deviennent inutiles en permettant la sat-règle. De plus, la fonction Flat rend les $\sqcap$- et $\sqcup$-règles inutiles. Enfin, la $\leq_r$-règle est utilisée pour identifier deux nœuds racines qui ne peuvent pas être gérés par la $\leq$-règle. Cette situation est illustrée dans l'exemple 13.

Exemple 13. Considérons une ontologie qui se compose des assertions suivantes :

$$(\leq 1R.C)(x), C(y), C(z), D(z), \exists S.D(z), R(x,y), R(x,z)$$

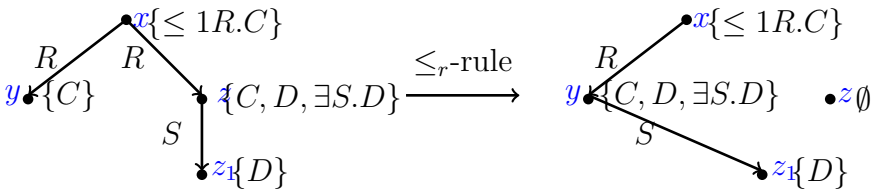


FIGURE 4.4 – Effet de la $\leq_r$-règle

Nous construisons une forêt de complétion en initialisant les nœuds racines x, y et z, et en ajoutant à leur étiquette les concepts imposés par les assertions $(\leq 1R.C)(x)$, $C(y)$, $C(z)$, $D(z)$ et $\exists S.D(z)$. Nous créons ensuite les arêtes entre x et y, et entre x et z avec l'étiquette R imposée par les assertions $R(x,y)$ et $R(x,z)$. En raison de la présence du concept $(\exists S.D)$ dans l'étiquette de z, La règle $\exists$-règle est appliquée pour générer un nouveau nœud z_1 qui est un S-successeur de z et possède un D dans son étiquette.

Lors de l'application de la $\leq$-règle pour la fusion de z à y, le changement $L(\langle x,z\rangle) = \emptyset$ est fait, et, donc, la relation R entre x et z ne sera pas conservée. Ceci implique que l'assertion $R(x,z)$ ne sera plus satisfaite. Pour cette raison, la $\leq_r$-règle introduit quelques étapes supplémentaires pour traiter ce problème. Premièrement, cette règle commence par ajouter $L(z)$ dans $L(y)$, comme ce que fait la $\leq$-règle, et puis par transformer les arêtes (et leurs étiquettes respectivement) entre z et ses voisins en celles (avec les étiquettes respectivement) entre y et ses voisins. Deuxièmement, $L(z)$ et toutes les arêtes allant de/à z sont supprimées de la forêt. Cette suppression ne cause aucun problème car tous les

voisins de z sont devenus voisins de y à l'étape précédente. Enfin, l'identification de z et y est sauvegardée dans la relation $\doteq$. Cette identification est nécessaire pour construire un tableau (modèle) à partir d'une forêt de complétion non-clash et complète.

Une autre question spécifique qui conduirait à la non-terminaison lorsque le raisonnement se fait avec une ABox est l'effet "yo-yo" [Baader and Sattler, 2001]. Ce problème se produit lors de la fusion de deux voisins y et z d'un nœud x (à cause d'un concept $\leq nR.C \in L(x)$) qui ne change pas la structure de voisinage de x qui déclenche la fusion. Pour résoudre ce problème, Horrocks *et al.* [Horrocks *et al.*, 2000] ont introduit la $\leq$-règle qui ajoute $L(z)$ dans $L(y)$ et fait $L(\langle x, z\rangle) = \emptyset$ au lieu de fusionner z dans y. Cette arête vide isole x de z et tous les successeurs de z qui sont responsables de la reproduction de la structure de voisinage de x.

Lemme 4. ***(Terminaison, Correction et Complétude).*** *Soit $\mathcal{O}$ une ontologie en $\mathcal{SHIQ}$.*

1. *L'algorithme de tableau avec les règles d'expansion de la figure 4.3 se termine.*
2. *Si l'algorithme de tableau avec les règles d'expansion de la figure 4.3 peut être appliqué à $\mathcal{O}$ de telle façon qu'il construise un arbre de complétion non-clash et complet alors $\mathcal{O}$ est cohérente ;*
3. *Si $\mathcal{O}$ est cohérente alors l'algorithme de tableau avec les règles d'expansion de la figure 4.3 peut être appliqué à $\mathcal{O}$ de telle façon qu'il construise un arbre de complétion non-clash et complet.*

Résumé de démonstration. La terminaison est une conséquence de la condition de blocage et de la monotonie de la construction d'une forêt de complétion $\mathcal{F}$, c'est-à-dire que l'algorithme ne supprime jamais rien de $\mathcal{F}$, sauf pour supprimer les arêtes par la $\leq_r$-règle, qui ne sont jamais restaurées. Pour prouver la correction, nous pouvons concevoir un modèle à partir d'une forêt de complétion $\mathcal{F}$ en "dénouage" ("unraveling" en anglais). Ce processus génère des descendants de nœuds bloqués en répliquant les descendants de nœuds de blocage. Le modèle obtenu est une forêt qui peut être une structure infinie. Pour démontrer la complétude, nous suivons la construction d'une forêt de complétion en utilisant une fonction π qui associe à chaque nœud x de la

forêt de complétion un individu du modèle. Chaque fois qu'une règle d'expansion non déterministe est appliquée à un nœud x, nous considérons les deux cas suivants : (i) elle fait un "bon choix" si $L(x) \subseteq L(\pi(x))$; (ii) elle fait un "mauvais choix" si $L(x) \nsubseteq L(\pi(x))$. Dans ce cas, nous montrons que l'algorithme peut toujours revenir en arrière pour faire un bon choix et réussir à construire une forêt de complétion non-clash et complète. Une preuve complète peut être trouvée en annexe. □

De façon similaire à la construction d'un arbre de complétion, l'algorithme de tableau peut construire une forêt de complétion dont la profondeur est limitée par une fonction exponentielle dans la taille de $\mathcal{O}$ en raison de la condition de blocage. Cela implique que la taille d'une forêt de complétion est limitée par une fonction doublement exponentielle de la taille de $\mathcal{O}$. Puisque l'algorithme crée une forêt de complétion d'une manière non-déterministe, il s'exécute en temps non-déterministe doublement exponentiel.

4.1.2 Modèle de forêt

Soit $\mathcal{O}$ une ontologie en $\mathcal{SHIQ}$, une forêt de complétion non-clash et complète $\mathcal{F}$ construite en appliquant l'algorithme de tableau avec les règles d'expansion de la figure 4.3 à $\mathcal{O}$ s'appelle un *modèle de forêt*. Selon la correction de l'algorithme de tableau (*cf.* lemme 4), pour chaque forêt de complétion non-clash et complète $\mathcal{F}$, il est possible d'obtenir un modèle désigné par $\mathcal{I}(\mathcal{F})$, par "dénouage" ("unraveling"). Dans ce cas, un nœud de $\mathcal{F}$ peut être reproduit pour un nombre infini d'individus du modèle. Étant donné un axiome $C \sqsubseteq D$, nous définissons $\mathcal{I}(\mathcal{F}) \models (C \sqsubseteq D)$ si $C^{\mathcal{I}(\mathcal{F})} \subseteq D^{\mathcal{I}(\mathcal{F})}$. Étant donné une assertion $C(a)$ (ou $R(a,b)$), nous définissons $\mathcal{I}(\mathcal{F}) \models C(a)$ (resp. $\mathcal{I}(\mathcal{F}) \models (R(a,b))$) si $a^{\mathcal{I}(\mathcal{F})} \in C^{\mathcal{I}(\mathcal{F})}$ (resp. $\langle a^{\mathcal{I}(\mathcal{F})}, b^{\mathcal{I}(\mathcal{F})} \rangle \in R^{\mathcal{I}(\mathcal{F})}$).

Inversement, selon la complétude de l'algorithme de tableau (*cf.* lemme 4), une forêt de complétion non-clash et complète $\mathcal{F}$ peut être construite à partir d'un modèle $\mathcal{I} \in \mathsf{Mod}(\mathcal{O})$, désigné par $\mathcal{F}(\mathcal{I})$.

Contrairement à l'algorithme de tableau classique qui se termine lorsque une forêt de complétion est construite avec succès, le nouvel algorithme de tableau doit considérer tous

les cas non-déterministes et construire tous les modèles de forêt pour $\mathcal{O}$.

Notation 2 ($\mathsf{FM}(\mathcal{O}, \mathsf{sub})$)**.** *Nous utilisons* $\mathsf{FM}(\mathcal{O})$ *pour désigner l'ensemble de tous les modèles de forêt pour une ontologie* $\mathcal{O}$ *en* $\mathcal{SHIQ}$*. Étant donné un ensemble de concepts* sub*, nous définissons* $\mathsf{FM}(\mathcal{O}, \mathsf{sub})$ *comme l'ensemble de tous les modèles de forêt construites en appliquant l'algorithme de tableau sur* $\mathcal{O}$ *tel que la* sat*-règle fonctionne sur* $\mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}$ *(i.e. elle sélectionne un sous-ensemble* $S \subseteq \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}$*). En particulier, étant donné une ontologie* $\mathcal{O}'$ *l'ensemble* $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ *peut être construit en exécutant l'algorithme de tableau sur* $\mathcal{O}$ *avec la* sat*-règle fonctionnant sur* $\mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$*. De plus, nous devons importer à* $\mathcal{O}$ *des rôles qui se trouvent dans les assertions de rôle de* $\mathcal{O}'$*. Cela conduit à ajouter à l'étiquette de chaque arête de racines* $\langle \widehat{x}, \widehat{y} \rangle$ *de chaque forêt* $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ *un sous-ensemble de rôles* $S_{\mathcal{R}} \subseteq \mathbf{R}_{\mathcal{O}'}$ *où* $\mathbf{R}_{\mathcal{O}'}$ *est l'ensemble de tous les rôles se trouvant dans* $\mathcal{O}'$ *avec leur inverse. Ce nouveau comportement peut être formalisé comme suit :*

— $\mathsf{sat}_{\mathcal{R}}$*-règle : pour chaque arête de racines* $\langle \widehat{x}, \widehat{y} \rangle \in \mathbf{E}$ *avec* $G = (\mathbf{V}, \mathbf{E}, \mathbf{L})$*,* $\mathcal{F} = (G, T\langle \widehat{x}_1 \rangle, \cdots, T\langle \widehat{x}_n \rangle)$ *et* $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$*, on ajoute* $L(\langle \widehat{x}, \widehat{y} \rangle) \leftarrow L(\langle \widehat{x}, \widehat{y} \rangle) \cup S_{\mathcal{R}}$ *pour certain* $S_{\mathcal{R}} \subseteq \mathbf{R}_{\mathcal{O}'}$*.*

Notons que la $\mathsf{sat}_{\mathcal{R}}$-règle n'est jamais appliquée en exécutant l'algorithme de tableau sur $\mathcal{O}$ pour construire $\mathsf{FM}(\mathcal{O})$. En revanche, lors de l'exécution de l'algorithme de tableau sur $\mathcal{O}$ pour construire $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$, elle est appliquée avec $S_{\mathcal{R}} \subseteq \mathbf{R}_{\mathcal{O}'}$.

La construction de $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ permet d'importer la signature d'une ontologie $\mathcal{O}'$ à $\mathcal{O}$ lors de la construction des forêts de complétion pour $\mathcal{O}$. Notons que l'on n'importe aucune contrainte sémantique de $\mathcal{O}'$ lors de la construction de $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$. Ce que nous réalisons réellement dans cette construction est l'importation dans $\mathcal{O}$ des concepts écrits dans la signature de $\mathcal{O}'$. Cette importation peut étendre $\mathsf{FM}(\mathcal{O})$ avec de nouvelles forêts de complétion mais jamais changer la cohérence de $\mathcal{O}$. L'effet de la $\mathsf{sat}_{\mathcal{R}}$-règle est illustré dans l'exemple 14.

Exemple 14. Considérons une ontologie $\mathcal{O}$ qui contient les axiomes et assertions

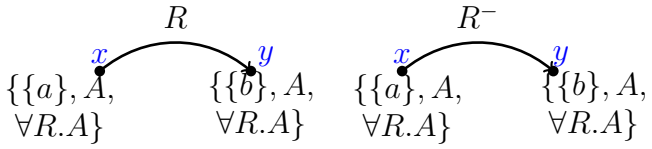
$$A \sqsubseteq \forall R.A, A(a), A(b)$$

Soit α une assertion $R(a, b)$, en appliquant l'algorithme de tableau sur $\mathcal{O}$ pour construire

l'ensemble $\mathsf{FM}(\mathcal{O})$, on obtient seulement la forêt $\mathcal{F}_1$ suivante :

$$\{\{a\}, A, \forall R.A\}^{x} \quad \{\{b\}, A, \forall R.A\}^{y}$$

En appliquant l'algorithme de tableau sur $\mathcal{O}$ avec la $\mathsf{sat}_{\mathcal{R}}$-règle fonctionnant sur $\mathbf{R}_\alpha = \{R, R^-\}$ pour construire l'ensemble $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\alpha))$, on obtient la forêt $\mathcal{F}_1$ ci-dessus et les deux forêts $\mathcal{F}_2$, $\mathcal{F}_3$ suivantes :



Alors, on obtient $\mathsf{FM}(\mathcal{O}) = \{\mathcal{F}_1\}$ mais $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\alpha)) = \{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3\}$.

Utilisons maintenant les notations récemment introduites pour formuler et prouver les propriétés suivantes sur $\mathsf{FM}(\mathcal{O})$ caractérisant la sémantique d'une ontologie $\mathcal{O}$.

Corollaire 2. *Soient $\mathcal{O}$ et $\mathcal{O}'$ deux ontologies cohérentes en $\mathcal{SHIQ}$, alors :*

1. *étant donné α un axiome ou assertion écrit en $\mathcal{S}(\mathcal{O})$, $\mathcal{I}(\mathcal{F}) \models \alpha$ pour chaque modèle de forêt $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\alpha))$ ssi $\mathcal{I} \models \alpha$ pour chaque modèle $\mathcal{I} \in \mathsf{Mod}(\mathcal{O})$;*
2. $\mathsf{FM}(\mathcal{O} \cup \mathcal{O}') = \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$.

Démonstration.

1. Supposons que pour chaque $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\alpha))$ on a $\mathcal{I}(\mathcal{F}) \models \alpha$ (*). Soit $\mathcal{I} \in \mathsf{Mod}(\mathcal{O})$ avec un domaine d'interprétation Δ. On doit montrer que $\mathcal{I} \models \alpha$. Pour chaque individu $a \in \Delta$ et pour chaque $C \in \mathsf{sub}(\alpha)$, on a $a \in C^{\mathcal{I}}$ ou $a \in (\neg C)^{\mathcal{I}}$. On peut modifier la construction de $\mathcal{F}(\mathcal{I})$ à partir de $\mathcal{I}$ lors de la démonstration de complétude de l'algorithme de tableau de telle manière que pour chaque nœud x de $\mathcal{F}(\mathcal{I})$ avec $\pi(x) = a$ (π est une fonction des nœuds de $\mathcal{F}(\mathcal{I})$ à Δ) et pour chaque $C \in \mathsf{sub}(\alpha)$ on ajoute C ou $\neg C$ à $L(x)$ si $a \in C^{\mathcal{I}}$ ou $a \in (\neg C)^{\mathcal{I}}$ respectivement. On désigne le $\mathcal{F}(\mathcal{I})$ modifié par $\mathcal{F}'(\mathcal{I})$. Selon la notation 2, $\mathcal{F}'(\mathcal{I}) \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\alpha))$. Par l'hypothèse (*), on obtient $\mathcal{I}(\mathcal{F}'(\mathcal{I})) \models \alpha$ (**).

   — $\alpha = C(d)$. Comme $\mathcal{I} \models C(d)$ ssi $\mathcal{O} \cup \{\neg C(d)\}$ incohérente, nous montrons que $\mathcal{O} \cup \{\neg C(d)\}$ est incohérente. En raisonnant par l'absurde, supposons que

$\mathcal{O} \cup \{\neg C(d)\}$ est incohérente. Cela implique qu'il y a un modèle $\mathcal{J} \in \mathsf{Mod}(\mathcal{O} \cup \{\neg C(d)\})$, *i.e.* $d \in (\neg C)^{\mathcal{J}}$. Par la construction de $\mathcal{F}'(\mathcal{J})$ à partir de $\mathcal{J}$, nous avons $\neg C \in L(x)$ où x est le nœud dans $\mathcal{F}'$ qui représente d. Par la construction de $\mathcal{I}(\mathcal{F}'(\mathcal{J}))$ à partir de $\mathcal{F}'(\mathcal{J})$, nous avons $d \in (\neg C)^{\mathcal{I}(\mathcal{F}'(\mathcal{J}))}$ (i).

D'autre part, nous avons $\mathcal{J} \in \mathsf{Mod}(\mathcal{O})$ car $\mathcal{J}$ satisfait une ontologie plus grande que $\mathcal{O}$. D'après (**), nous obtenons $\mathcal{I}(\mathcal{F}'(\mathcal{J})) \models C(d)$, *i.e.* $d \in (C)^{\mathcal{I}(\mathcal{F}'(\mathcal{J}))}$, qui contredit (i).

— $\alpha = R(d, d')$. Comme $\mathcal{I} \models R(d, d')$ ssi $\mathcal{O} \cup \{\neg R(d, d')\}$ incohérente, nous montrons que $\mathcal{O} \cup \{\neg R(d, d')\}$ est incohérente. En raisonnant par l'absurde, supposons que $\mathcal{O} \cup \{\neg R(d, d')\}$ est cohérente. Cela implique qu'il y a un modèle $\mathcal{J} \in \mathsf{Mod}(\mathcal{O} \cup \{\neg R(d, d')\})$, *i.e.* $\langle d, d' \rangle \notin R^{\mathcal{J}}$. Par la construction de $\mathcal{F}'(\mathcal{J})$ à partir de $\mathcal{J}$, nous avons $R \notin L(\langle x, y \rangle)$ où x, y sont les nœuds dans $\mathcal{F}'$ qui représentent d, d' respectivement. Par la construction de $\mathcal{I}(\mathcal{F}'(\mathcal{J}))$ à partir de $\mathcal{F}'(\mathcal{J})$, nous avons $\langle d, d' \rangle \notin R^{\mathcal{I}(\mathcal{F}'(\mathcal{J}))}$ (ii).

D'autre part, nous avons $\mathcal{J} \in \mathsf{Mod}(\mathcal{O})$ car $\mathcal{J}$ satisfait une ontologie plus grande que $\mathcal{O}$. D'après (**), nous obtenons $\mathcal{I}(\mathcal{F}'(\mathcal{J})) \models R(d, d')$, *i.e.* $\langle d, d' \rangle \in R^{\mathcal{I}(\mathcal{F}'(\mathcal{J}))}$, qui contredit (ii).

— $\alpha = (C \sqsubseteq D)$. Comme $\mathcal{I} \models C \sqsubseteq D$ ssi $\mathcal{O} \cup \{(C \sqcap \neg D)(d)\}$ incohérente pour un certain nouvel individu d, on montre que $\mathcal{O} \cup \{(C \sqcap \neg D)(d)\}$ est incohérente. Supposons que $\mathcal{O} \cup \{(C \sqcap \neg D)(d)\}$ est cohérente. Cela implique qu'il y a un modèle $\mathcal{J} \in \mathsf{Mod}(\mathcal{O} \cup \{(C \sqcap \neg D)(d)\})$, *i.e.* $d \in ((C \sqcap \neg D))^{\mathcal{J}}$. Par la construction de $\mathcal{F}'(\mathcal{J})$ à partir de $\mathcal{J}$, on a $(C \sqcap \neg D) \in L(x)$ où x est un nœud dans $\mathcal{F}'$ qui représente d. Par la construction de $\mathcal{I}(\mathcal{F}'(\mathcal{J}))$ à partir de $\mathcal{F}'(\mathcal{J})$, on a $d \in (C \sqcap \neg D)^{\mathcal{I}(\mathcal{F}'(\mathcal{J}))}$ (iii).

D'autre part, on a $\mathcal{J} \in \mathsf{Mod}(\mathcal{O})$ car $\mathcal{J}$ satisfait une ontologie plus grande que $\mathcal{O}$. Selon (**), on obtient $\mathcal{I}(\mathcal{F}'(\mathcal{J})) \models (C \sqsubseteq D)$, *i.e.* $C^{\mathcal{I}(\mathcal{F}'(\mathcal{J}))} \subseteq D^{\mathcal{I}(\mathcal{F}'(\mathcal{J}))}$, ce qui contredit (iii).

Inversement, supposons que pour chaque $\mathcal{I} \in \mathsf{Mod}(\mathcal{O})$ on a $\mathcal{I} \models \alpha$. Soit $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\alpha))$. On doit démontrer que $\mathcal{I}(\mathcal{F}) \models \alpha$. Ce qui est simple puisque $\mathcal{I}(\mathcal{F}) \in \mathsf{Mod}(\mathcal{O})$.

2. D'abord, démontrons que $\mathsf{FM}(\mathcal{O} \cup \mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$. Soit $\mathcal{F} \in \mathsf{FM}(\mathcal{O} \cup \mathcal{O}')$. Comme $\mathcal{F}$ est construite en appliquant l'algorithme de tableau sur $\mathcal{O} \cup \mathcal{O}'$, chaque axiome et assertion dans $\mathcal{O} \cup \mathcal{O}'$ est satisfait dans chaque étiquette de nœud et d'arc de $\mathcal{F}$, et la sat-règle fonctionne sur $\mathsf{sub}(\mathcal{O} \cup \mathcal{O}')$. Cela implique que l'on peut reconstruire $\mathcal{F}$ en appliquant l'algorithme de tableau sur $\mathcal{O}$ avec la sat-règle fonctionnant sur $\mathsf{sub}(\mathcal{O} \cup \mathcal{O}')$. Cela veut dire que $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$. De la même manière, $\mathcal{F} \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$. Donc, $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$. D'où, $\mathsf{FM}(\mathcal{O} \cup \mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ (i).

   On démontre que $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \subseteq \mathsf{FM}(\mathcal{O} \cup \mathcal{O}')$. Soit $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$. Comme $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$, chaque axiome et assertion dans $\mathcal{O}$ et $\mathcal{O}'$ est satisfait dans chaque étiquette de nœud et d'arc de $\mathcal{F}$, et la sat-règle fonctionne sur $\mathsf{sub}(\mathcal{O} \cup \mathcal{O}')$. Cela implique que l'on peut reconstruire $\mathcal{F}$ en appliquant l'algorithme de tableau sur $\mathcal{O} \cup \mathcal{O}'$ avec la sat-règle fonctionnant sur $\mathsf{sub}(\mathcal{O} \cup \mathcal{O}')$; et que $\mathcal{F} \in \mathsf{FM}(\mathcal{O} \cup \mathcal{O}')$. Donc, $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \subseteq \mathsf{FM}(\mathcal{O} \cup \mathcal{O}')$ (ii).

   Selon (i) et (ii), on a $\mathsf{FM}(\mathcal{O} \cup \mathcal{O}') = \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$.

□

Le corollaire 2 confirme l'équivalence de sémantique entre $\mathsf{Mod}(\mathcal{O})$ et $\mathsf{FM}(\mathcal{O})$ dans le sens où $\mathsf{FM}(\mathcal{O})$ peut représenter $\mathsf{Mod}(\mathcal{O})$. Ce résultat nous permet de remplacer un ensemble éventuellement infini $\mathsf{Mod}(\mathcal{O})$ par un ensemble fini $\mathsf{FM}(\mathcal{O})$ grâce aux constructions présentées dans les sections suivantes.

4.2 Opération de révision

Rappelons que notre objectif principal est de définir une opération de révision qui permet de réviser une ontologie cohérente $\mathcal{O}$ par des axiomes ou assertions d'une autre ontologie cohérente $\mathcal{O}'$, mais où $\mathcal{O} \cup \mathcal{O}'$ est incohérente. Une telle opération de révision renvoie un ensemble de forêts de complétion dont une ontologie de révision devrait admettre pour prendre en compte les nouvelles connaissances de $\mathcal{O}'$, et être sémantiquement aussi proche

que possible de $\mathcal{O}$. Pour atteindre cet objectif, nous devons introduire une distance entre deux forêts de complétion, ce qui donne un pré-ordre total sur elles et permet de parler de similarité entre deux ontologies. Cette distance est une extension de celle définie sur les arbres de complétion vus dans le chapitre précédent. Enfin, nous montrerons que notre opération de révision satisfait tous les postulats d'AGM. Mais commençons cette section en étendant les isomorphismes définis sur les arbres de complétion à ceux définis sur les forêts de complétion.

4.2.1 Distance entre des modèles de forêt

Définition 11 (**Isomorphisme**). *Soient* $\mathcal{F}=(G, T\langle\widehat{x}_1\rangle, \ldots, T\langle\widehat{x}_n\rangle)$ *et* $\mathcal{F}' = (G', T\langle\widehat{x}'_1\rangle, \ldots, T\langle\widehat{x}'_n\rangle)$ *deux modèles de forêt avec* $G = (\mathbf{V}, \mathbf{E}, \mathbf{L})$, $G' = (\mathbf{V}', \mathbf{E}', \mathbf{L}')$, $T\langle\widehat{x}_i\rangle = \langle V_i, L_i, E_i\rangle$ *et* $T\langle\widehat{x}'_j\rangle = \langle V'_j, L'_j, E'_j\rangle$ $(1 \leq i, j \leq n)$, *soient* $\mathcal{V} = \mathbf{V} \cup V_1 \cup \cdots \cup V_n$ *et* $\mathcal{V}' = \mathbf{V}' \cup V'_1 \cup \cdots \cup V'_n$, *soient* $\mathcal{E} = \mathbf{E} \cup E_1 \cup \cdots \cup E_n$ *et* $\mathcal{E}' = \mathbf{E}' \cup E'_1 \cup \cdots \cup E'_n$. *Nous utilisons* $\mathsf{succ}(x)$ *pour désigner l'ensemble des successeurs d'un nœud* x *dans un arbre* $T\langle\widehat{x}_i\rangle$ *ou* $T\langle\widehat{x}'_j\rangle$ *avec* $1 \leq i, j \leq n$.

- $T\langle\widehat{x}_i\rangle$ *et* $T\langle\widehat{x}'_j\rangle$ *sont* isomorphes *pour* $1 \leq i, j \leq n$ *s'il y a une bijection* π *de* V_i *à* V'_j *telle que (i)* $\pi(\widehat{x}_i) = \widehat{x}'_j$ *et (ii) pour chaque nœud* $x \in V_i$, *nous avons* $\pi(x') \in \mathsf{succ}(\pi(x))$ *pour chaque* $x' \in \mathsf{succ}(x)$.
- $\mathcal{F}$ *et* $\mathcal{F}'$ *sont isomorphes s'il y a une bijection* π *de* $\mathcal{V}$ *à* $\mathcal{V}'$ *telle que*
  - $\pi(\widehat{x}_i) = \widehat{x}'_j$ *pour chaque* $\widehat{x}_i \in \mathbf{V}$,
  - *pour chaque* $T\langle\widehat{x}_i\rangle \in \mathcal{F}$, *deux arbres* $T\langle\widehat{x}_i\rangle$ *et* $T\langle\pi(\widehat{x}_i)\rangle$ *sont* isomorphes.

  Dans ce cas, π *est dit isomorphisme entre* $\mathcal{F}$ *et* $\mathcal{F}'$.
- $\mathcal{F}$ *et* $\mathcal{F}'$ *sont équivalentes s'il existe un isomorphisme* π *entre* $\mathcal{F}$ *et* $\mathcal{F}'$ *tel que* $L(x) = L'(\pi(x))$ *pour chaque* $x \in V$, *et* $L(\langle x, y\rangle) = L'(\langle\pi(x), \pi(y)\rangle)$ *pour chaque* $\langle x, y\rangle \in E$ *où* $L(x) = L_i(x)$ *(resp.* $L(x) = L'_j(x)$*) si* $x \in V_i$ *(resp.* $x \in V'_j$*), et* $L(\widehat{x}_i) = \mathbf{L}(\widehat{x}_i)$ *(resp.* $L(\widehat{x}_i) = \mathbf{L}'(\widehat{x}'_j)$*) si* $\widehat{x}_i \in \mathbf{V}$ *(resp.* $\widehat{x}'_j \in \mathbf{V}'$*).*

Notons que s'il existe une bijection π entre deux arbres $T\langle\widehat{x}_i\rangle$ et $T\langle\widehat{x}'_j\rangle$ comme décrite dans la définition 11 alors la restriction de π à $\mathsf{succ}(x)$, désignée par $\pi|_{\mathsf{succ}(x)}$, est une bijection de

$\mathsf{succ}(x)$ à $\mathsf{succ}(\pi(x))$. En effet, $\pi|_{\mathsf{succ}(x)}$ est une injection de $\mathsf{succ}(x)$ à $\mathsf{succ}(\pi(x))$ en raison du premier point de la définition 11. Il affirme que $\pi|_{\mathsf{succ}(x)}$ est aussi une surjection de $\mathsf{succ}(x)$ à $\mathsf{succ}(\pi(x))$ car s'il existe certains $y \in \mathsf{succ}(\pi(x))$ tels que $y = \pi(y')$ pour certains $y' \in \mathsf{succ}(x')$ avec $x' \neq x$ alors $\pi(y') \in \mathsf{succ}(\pi(x'))$ à cause du premier point de la définition 11. Ceci implique que $\pi(y') \in \mathsf{succ}(\pi(x)) \cap \mathsf{succ}(\pi(x')) = \emptyset$, ce qui n'est pas possible.

Remarque 1. *Soient $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ et $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ deux modèles de forêt avec les ensembles de nœuds racines $\mathbf{V}$ et $\mathbf{V}'$. Il est nécessaire d'importer tous les individus de $\mathcal{O}$ (inclus dans $\mathsf{sub}(\mathcal{O})$) à $\mathcal{O}'$ et inversement lors de la révision de $\mathcal{O}$ par de nouveaux axiomes de $\mathcal{O}'$. D'où, pour chaque individu a il y a un seul arbre $T\langle\widehat{x}_i\rangle$ de $\mathcal{F}$ et un seul arbre $T\langle\widehat{x}'_j\rangle$ de $\mathcal{F}'$ tels que $a \in L(\widehat{x}_i) \cap L(\widehat{x}'_j)$. Cela implique qu'il existe une bijection φ de $\mathbf{V}$ à $\mathbf{V}'$ telle que $\varphi(\widehat{x}_i) = \widehat{x}'_j$ ssi $a \in L(\widehat{x}_i) \cap L(\widehat{x}'_j)$ pour un certain individu a.*

Puisque la notion d'isomorphisme ne se réfère qu'à la structure des forêts de complétion, on peut toujours obtenir un tel isomorphisme entre deux forêts de complétion en ajoutant des nœuds et des arêtes vides à ces dernières. Ceci est similaire à ce que nous avons fait entre deux arbres de complétion dans le chapitre précédent. Dans ce qui suit, nous introduisons une distance entre deux forêts de complétion isomorphes. Afin d'obtenir la différence entre deux forêts de complétion de même structure, cette distance fait référence à l'étiquette des nœuds et des arêtes dans les forêts de complétion et s'appuie sur l'opérateur de différence symétrique, désigné par $\triangle$, défini comme suit : $S \triangle S' = (S \cup S') \setminus (S \cap S')$ pour deux ensembles S et S'.

Définition 12 (Distance). *Soient $\mathcal{F}=(G, T\langle\widehat{x}_1\rangle, \ldots, T\langle\widehat{x}_n\rangle)$ et $\mathcal{F}' = (G', T\langle\widehat{x}'_1\rangle, \ldots, T\langle\widehat{x}'_n\rangle)$ deux modèles de forêt avec $G = (\mathbf{V}, \mathbf{E}, \mathbf{L})$, $G' = (\mathbf{V}', \mathbf{E}', \mathbf{L}')$, $T\langle\widehat{x}_i\rangle = \langle V_i, L_i, E_i\rangle$ et $T\langle\widehat{x}'_j\rangle = \langle V'_j, L'_j, E'_j\rangle$ pour $1 \leq i, j \leq n$. Soit φ une bijection de $\mathbf{V}$ à $\mathbf{V}'$ telle que $\varphi(\widehat{x}_i) = \widehat{x}'_j$ ssi il y a un individu a satisfaisant $a \in \mathbf{L}(\widehat{x}_i) \cap \mathbf{L}'(\widehat{x}'_j)$. La distance entre $\mathcal{F}$ et $\mathcal{F}'$, désignée par $d(\mathcal{F}, \mathcal{F}')$, est définie comme suit :*

$$d(\mathcal{F}, \mathcal{F}') = \sum_{i=1}^{n} d(T\langle\widehat{x}_i\rangle, T\langle\varphi(\widehat{x}_i)\rangle) \;+\; \max_{\langle x,y\rangle \in \mathbf{E}} (|L(\langle x, y\rangle) \triangle L'(\langle\varphi(x), \varphi(y)\rangle)|)$$

$$\text{où } d(T, T') = \min_{\pi \in \Pi(T,T')} \{ \max_{\langle x,y\rangle \in E} (|L(x) \triangle L'(\pi(x))| + |L(\langle x, y\rangle) \triangle L'(\langle\pi(x), \pi(y)\rangle)| + \\ + |L(y) \triangle L'(\pi(y))|)\}$$

et $\Pi(T, T')$ est l'ensemble de tous les isomorphismes entre deux arbres T et T'.

Nous avons défini une distance sur des modèles de forêt dont les ensembles de nœuds racines doivent être associés par une bijection φ d'après la remarque 1. Cette restriction est justifiée par le fait que (i) les étiquettes de deux nœuds seraient comparées contenant le même individu plutôt que contenant deux individus différents et (ii) contrairement à un concept qui peut représenter un ensemble d'individus et se trouve plusieurs fois dans un modèle de forêt, un nom d'individu se trouve une seule fois dans un modèle de forêt.

Comme toute distance, $d(\mathcal{F}, \mathcal{F}')$ devrait permettre de mesurer la différence entre deux forêts $\mathcal{F}$ et $\mathcal{F}'$, c'est-à-dire, elle devrait satisfaire les propriétés d'identité, de symétrie et d'inégalité triangulaire. Pour cela, nous utilisons un opérateur, à savoir max, qui représente la plus grande différence entre deux triplets (chacun composé d'une arête et deux nœuds) associés par un isomorphisme π entre $\mathcal{F}$ et $\mathcal{F}'$. Cette opération max nous permet de garantir la propriété d'identité mais elle n'est pas suffisante pour garantir la propriété d'inégalité triangulaire $d(\mathcal{F}, \mathcal{F}') \leq d(\mathcal{F}, \mathcal{F}'') + d(\mathcal{F}'', \mathcal{F}')$. Pour cette raison, nous devons utiliser un autre opérateur, à savoir min, qui permet de choisir un isomorphisme π à partir de tous les isomorphismes entre $\mathcal{F}$ et $\mathcal{F}'$ (dont l'un participe à la détermination de $d(\mathcal{F}, \mathcal{F}'') + d(\mathcal{F}'', \mathcal{F}')$) tel que la plus grande différence entre les triplets associés par π est la plus petite. De plus, calculer la distance en utilisant directement la définition 12 peut conduire à une explosion exponentielle car il peut y avoir un nombre exponentiel de différents isomorphismes entre deux forêts de complétion.

Lemme 5. *La fonction $d(\mathcal{F}, \mathcal{F}')$ dans la définition 12 satisfait les propriétés d'identité, de symétrie et d'inégalité triangulaire.*

Résumé de démonstration. La propriété de symétrie est satisfaite en raison de la commutativité de l'opérateur de différence symétrique $\triangle$ sur deux ensembles. La propriété d'identité est une conséquence de la propriété $S \triangle S' = \emptyset$ ssi $S = S'$ et l'opérateur max sur chaque terme $L(\langle x, y\rangle) \triangle L(\langle \varphi(x), \varphi(y)\rangle)$. L'opérateur min qui se réfère à l'ensemble de tous les isomorphismes entre deux forêts comparables et la transitivité des isomorphismes traversant les modèles de forêt aident à établir l'inégalité triangulaire. En effet, pour montrer $d(\mathcal{F}, \mathcal{F}') \leq d(\mathcal{F}, \mathcal{F}'') + d(\mathcal{F}'', \mathcal{F}')$, nous prenons un isomorphisme π_1 de $\Pi(\mathcal{F}, \mathcal{F}'')$ tel que π_1 donne $d(\mathcal{F}, \mathcal{F}'')$, et un isomorphisme π_2 de $\Pi(\mathcal{F}'', \mathcal{F}')$ tel que π_2 donne $d(\mathcal{F}'', \mathcal{F}')$. À partir de π_1 et π_2, nous

pouvons définir un isomorphisme $\pi_2 \circ \pi_1$ de $\mathcal{F}$ à $\mathcal{F}'$ qui devrait donner une valeur supérieure à $d(\mathcal{F}, \mathcal{F}')$. Une preuve complète de ce lemme se trouve en annexe. □

Montrons maintenant que la distance de la définition 12 donne un pré-ordre total sur l'ensemble des forêts de complétion isomorphes. Pour ce faire, nous définissons une relation "$\mathcal{F} \leq \mathcal{F}'$" sur des forêts de complétion isomorphes incluant une forêt $\mathcal{F}_0$ qui contient seulement des étiquettes vides comme suit : $\mathcal{F} \leq \mathcal{F}'$ ssi $d(\mathcal{F}_0, \mathcal{F}) \leq d(\mathcal{F}_0, \mathcal{F}')$.

Lemme 6. *La relation "$\leq$" est un pré-ordre total sur des forêts de complétion isomorphes.*

Démonstration. Soient $\mathcal{F}, \mathcal{F}', \mathcal{F}''$ et $\mathcal{F}_0$ des forêts de complétion isomorphes où $\mathcal{F}_0$ a des étiquettes vides. Nous devons prouver les propriétés suivantes :

1. **Transitivité :** par la définition, $\mathcal{F} \leq \mathcal{F}'$ implique $d(\mathcal{F}_0, \mathcal{F}) \leq d(\mathcal{F}_0, \mathcal{F}')$, et $\mathcal{F}' \leq \mathcal{F}''$ implique $d(\mathcal{F}_0, \mathcal{F}') \leq d(\mathcal{F}_0, \mathcal{F}'')$. Donc, $d(\mathcal{F}_0, \mathcal{F}) \leq d(\mathcal{F}_0, \mathcal{F}'')$. Par la définition, nous obtenons $\mathcal{F} \leq \mathcal{F}''$. Par conséquent, nous avons que si $\mathcal{F} \leq \mathcal{F}'$ et $\mathcal{F}' \leq \mathcal{F}''$ alors $\mathcal{F} \leq \mathcal{F}''$.
2. **Totalité :** d'après la définition 12 et le lemme 5, nous avons toujours $d(\mathcal{F}_0, \mathcal{F}) \leq d(\mathcal{F}_0, \mathcal{F}')$ ou $d(\mathcal{F}_0, \mathcal{F}') \leq d(\mathcal{F}_0, \mathcal{F})$. Ceci implique $\mathcal{F} \leq \mathcal{F}'$ ou $\mathcal{F}' \leq \mathcal{F}$.
3. **Réflexivité :** d'après la définition 12 et le lemme 5, nous avons $d(\mathcal{F}_0, \mathcal{F}) \leq d(\mathcal{F}_0, \mathcal{F})$. D'où, $\mathcal{F} \leq \mathcal{F}$.

□

4.2.2 Opération de révision fondée sur modèles de forêt

Toutes les notions précédentes fournissent suffisamment d'éléments pour définir une opération de révision pour une ontologie $\mathcal{O}$ en $\mathcal{SHIQ}$ par une autre ontologie $\mathcal{O}'$. Cette opération détermine un ensemble de modèles qu'une ontologie de révision doit admettre. De plus, le corollaire 2 nous permet de représenter la sémantique de $\mathcal{O}$ en utilisant un ensemble fini $\mathsf{FM}(\mathcal{O})$ au lieu d'un ensemble éventuellement infini $\mathsf{Mod}(\mathcal{O})$.

Définition 13 (Opération de révision). *Soient $\mathcal{O}$ et $\mathcal{O}'$ deux ontologies cohérentes en $\mathcal{SHIQ}$. Un ensemble de modèles de forêt de la révision de $\mathcal{O}$ par $\mathcal{O}'$, désigné par $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$, est défini comme suit :*

$$\mathsf{FM}(\mathcal{O},\mathcal{O}') = \{\mathcal{F} \in \mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O})) \mid \exists \mathcal{F}_0 \in \mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}')),$$
$$\forall \mathcal{F}' \in \mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O})), \mathcal{F}'' \in \mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}')) : d(\mathcal{F},\mathcal{F}_0) \leq d(\mathcal{F}',\mathcal{F}'')\}$$

Intuitivement, parmi les modèles de forêt dans $\mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O}))$, $\mathsf{FM}(\mathcal{O},\mathcal{O}')$ ne conserve que les modèles de forêt qui sont les plus proches de ceux dans $\mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}'))$ grâce à l'opération $d(\mathcal{F}_1,\mathcal{F}_2)$ qui caractérise la différence entre $\mathcal{F}_1$ et $\mathcal{F}_2$.

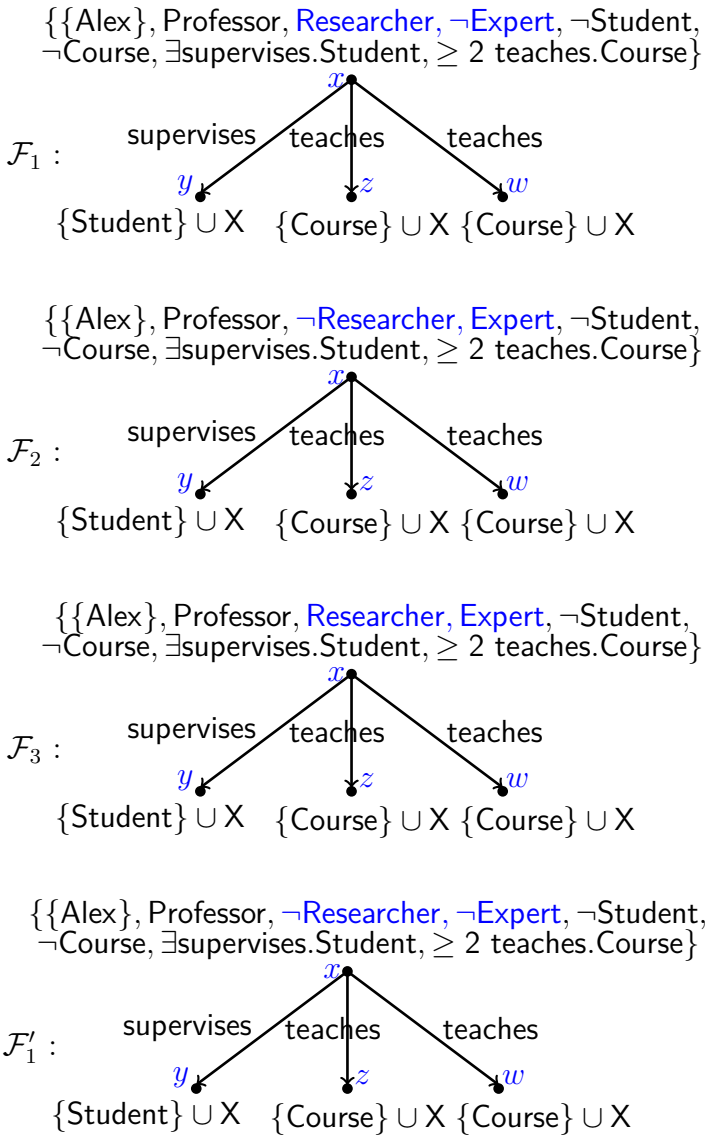
Exemple 15. Reprenons l'ontologie UNI de l'exemple 12. Pour la simplification, supposons que $\mathcal{O}$ est une ontologie obtenue en ajoutant dans UNI les axiomes du tableau 4.3 et $\mathcal{O}'$ consiste en δ_1, δ_2. En appliquant le nouvel algorithme de tableau sur $\mathcal{O}$, l'ensemble $\mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}'))$ contient 3 modèles de forêt $\mathcal{F}_1, \mathcal{F}_2$, et $\mathcal{F}_3$ dans la figure 4.5. En appliquant le nouvel algorithme de tableau sur $\mathcal{O}'$, nous obtenons un modèle de forêt $\mathcal{F}'_1 \in \mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O}))$ illustré par la figure 4.5 parmi d'autres modèles de forêt. En appliquant la distance introduite dans la définition 12, nous obtenons $d(\mathcal{F}'_1,\mathcal{F}_3) = 4$ et $d(\mathcal{F}'_1,\mathcal{F}_1) = d(\mathcal{F}'_1,\mathcal{F}_2) = 2$. D'après la 15, $\mathsf{FM}(\mathcal{O},\mathcal{O}')$ contient un seul modèle de forêt $\mathcal{F}'_1$. Notons que $d(\mathcal{F}'_x,\mathcal{F}_y) > 2$ pour toutes $\mathcal{F}_y \in \mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}'))$ et $\mathcal{F}'_x \in \mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O}))$ avec $\mathcal{F}'_x \neq \mathcal{F}'_1$.

α_4 : $\neg\mathsf{Professor} \sqsubseteq \forall \mathsf{supervises}.(\neg\mathsf{Student}) \sqcap (\leq 1\ \mathsf{teaches}.\mathsf{Course})$	Quelqu'un qui n'est pas un professeur ne peut pas superviser un étudiant et n'enseigne pas plus d'un cours.
α_5 : $\mathsf{Student} \sqcap \mathsf{Course}, \mathsf{Student} \sqcap \mathsf{Professor}, \mathsf{Student} \sqcap \mathsf{Researcher}, \mathsf{Student} \sqcap \mathsf{Expert}$	Un étudiant n'est ni un cours, ni un professeur, ni un chercheur, ni un .
α_6 : $\mathsf{Course} \sqcap \mathsf{Professor}, \mathsf{Course} \sqcap \mathsf{Researcher}, \mathsf{Course} \sqcap \mathsf{Expert}$	Un cours n'est ni un professeur, ni un chercheur, ni un expert.

TABLEAU 4.3 – Axiomes ajoutés dans l'ontologie UNI

4.2.3 Satisfaction des Postulats d'AGM

Rappelons que notre objectif est de proposer une opération de révision qui assure le principe de changement minimal introduit par Alchourrón, Gärdenfors et Makinson [Alchourrón *et al.*, 1985] comme les postulats d'AGM. Comme mentionné dans le chapitre 2, Katsuno et Mendelzon [Katsuno and Mendelzon, 1991] ont reformulé les postulats d'AGM pour des bases de connaissances en logique propositionnelle, à savoir **(R∘1)**-**(R∘6)** et mon-



où X = {¬Professor, ¬Expert, ¬Researcher, ∀supervises.(¬Student), (≤ 1 teaches.Course)}

FIGURE 4.5 – Forêts de complétion donnant les modèles d'UNI ($\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$) et de $\{\delta_1, \delta_2\}$ ($\mathcal{F}'_1$)

tré que l'existence d'un pré-ordre total sur les modèles d'une base de connaissances est équivalente à **(R∘1)**-**(R∘6)**.

L'opération de révision selon la définition 13 fournit directement la satisfaction du principe du changement minimal. Effectivement, $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$ ne conserve que des modèles de forêt de $\mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ les plus proches de ceux de $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ d'après la distance entre des forêts de complétion. Cette distance infère le pré-ordre total "$\leq$" sur des modèles de forêt. Cette observation nous permet d'obtenir directement le résultat disant que les postulats de révision impliquent un pré-ordre total sur des modèles de forêt car nous considérons

seulement des modèles de forêt sur lesquels un pré-ordre total existe déjà. Il reste à prouver que l'opération de révision de la définition 13 satisfait les postulats de révision. Pour ce faire, nous utilisons les postulats **(Q1)**-**(Q6)** reformulés par Qi, Liu et Bell [Qi and Du, 2009] pour des ontologies en LD, comme présentés dans le chapitre 3.

D'après le corollaire 2 et la définition 13, nous pouvons remplacer $\mathsf{Mod}(\mathcal{O})$ et $\mathsf{Mod}(\mathcal{O} \circ \mathcal{O}')$ avec $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ et $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$, respectivement, pour obtenir des postulats de révision qui ne se réfèrent qu'à des structures calculables. Nous reformulons maintenant les postulats de révision dans notre contexte comme suit.

(P1) $\mathcal{I}(\mathcal{F}) \models \alpha$ pour chaque $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathcal{O}')$ et chaque axiome $\alpha \in \mathcal{O}'$;

(P2) Si $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \neq \emptyset$,
alors $\mathsf{FM}(\mathcal{O}, \mathcal{O}') = \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$;

(P3) Si $\mathcal{O}'$ est cohérente alors $\mathsf{FM}(\mathcal{O}, \mathcal{O}') \neq \emptyset$;

(P4) Si $\mathsf{FM}(\mathcal{O}_1, \mathsf{sub}(\mathcal{O}_1')) = \mathsf{FM}(\mathcal{O}_2, \mathsf{sub}(\mathcal{O}_2'))$ et
$\mathsf{FM}(\mathcal{O}_1', \mathsf{sub}(\mathcal{O}_1)) = \mathsf{FM}(\mathcal{O}_2', \mathsf{sub}(\mathcal{O}_2))$, alors $\mathsf{FM}(\mathcal{O}_1, \mathcal{O}_1') = \mathsf{FM}(\mathcal{O}_2, \mathcal{O}_2')$;

(P5) $\mathsf{FM}(\mathcal{O}, \mathcal{O}') \cap \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')) \subseteq \mathsf{FM}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'')$;

(P6) Si $\mathsf{FM}(\mathcal{O}, \mathcal{O}') \cap \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')) \neq \emptyset$,
alors $\mathsf{FM}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'') \subseteq \mathsf{FM}(\mathcal{O}, \mathcal{O}') \cap \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}'))$.

Intuitivement, **(P1)** garantit que tous les axiomes de la nouvelle ontologie $\mathcal{O}'$ peuvent être déduits à partir du résultat de révision. **(P2)** énonce que l'ontologie initiale $\mathcal{O}$ n'est pas changée s'il n'y a aucun conflit. **(P3)** est une condition empêchant une révision de l'incohérence injustifiée. **(P4)** spécifie que l'opération de révision doit être indépendante de la syntaxe des ontologies. En effet, si $\mathcal{O}_1$ et $\mathcal{O}_1'$ sont remplacées par $\mathcal{O}_2$ et $\mathcal{O}_2'$ telles qu'elles admettent les mêmes modèles de forêt alors les ontologies de révision admettent aussi les mêmes modèles de forêt. Quant aux **(P5)** et **(P6)**, nous utilisons la figure 4.6 pour illustrer comment ils peuvent garantir le principe de changement minimal. Par souci de concision, nous utiliserons des ontologies $\mathcal{O}, \mathcal{O}'$ et $\mathcal{O}''$ telles que $\mathsf{sub}(\mathcal{O}) = \mathsf{sub}(\mathcal{O}') = \mathsf{sub}(\mathcal{O}'')$. Dans ce cas, nous pouvons écrire $\mathsf{FM}(\mathcal{X})$ pour $\mathsf{FM}(\mathcal{X}, \mathsf{sub}(\mathcal{X}'))$ où $\mathcal{X}, \mathcal{X}' \in \{\mathcal{O}, \mathcal{O}', \mathcal{O}''\}$. La partie rayée de $\mathsf{FM}(\mathcal{O}')$ apparaissant dans la figure 4.6 représente $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$ car $\mathsf{FM}(\mathcal{O}, \mathcal{O}') \subseteq$

$\mathsf{FM}(\mathcal{O}')$ selon la définition 13. Si le principe de changement minimal n'est pas garanti, il existe une $\mathcal{F}_0 \in \mathsf{FM}(\mathcal{O}') \setminus \mathsf{FM}(\mathcal{O}, \mathcal{O}')$ telle que $\mathcal{F}_0$ soit plus proche de $\mathsf{FM}(\mathcal{O})$ qu'une autre $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}, \mathcal{O}')$. Puisque **(P5)** et **(P6)** sont satisfaits pour toute ontologie $\mathcal{O}''$, nous pouvons prendre $\mathcal{O}''$ telle que $\mathcal{F}_0 \in \mathsf{FM}(\mathcal{O}'')$. Cela implique que $\mathcal{F}_0$ doit appartenir à $\mathsf{FM}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'')$ car $\mathcal{F}_0 \in \mathsf{FM}(\mathcal{O}' \cup \mathcal{O}'')$ et elle est la plus proche de $\mathsf{FM}(\mathcal{O})$ parmi celles appartenant à $\mathsf{FM}(\mathcal{O}' \cup \mathcal{O}'')$. Cependant, **(P5)** et **(P6)** valident $\mathsf{FM}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'') = \mathsf{FM}(\mathcal{O}, \mathcal{O}') \cap \mathsf{FM}(\mathcal{O}'')$. Cela force $\mathcal{F}_0$ à devenir une certaine $\mathcal{F}'_0 \in \mathsf{FM}(\mathcal{O}, \mathcal{O}')$, ce qui est contradictoire.

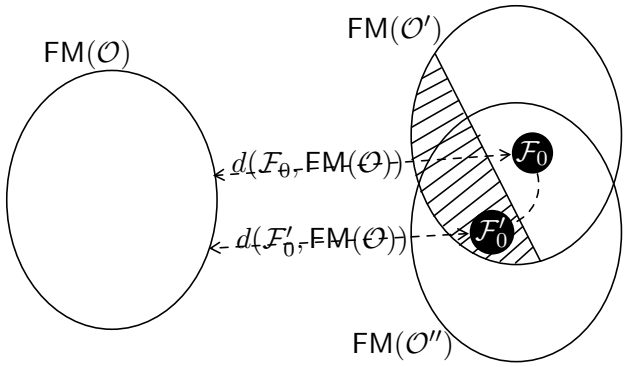


FIGURE 4.6 – **(P5)** & **(P6)** garantissent le principe de changement minimal

Nous pouvons prouver que, dans notre cadre de travail, tous les postulats sont toujours respectés lors de la révision d'ontologies en $\mathcal{SHIQ}$ avec individus.

Théorème 4. *L'opération de révision* $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$ *décrite dans la définition 13 satisfait les postulats* ***(P1)-(P6)****.*

Résumé de démonstration. **(P1)** : il peut être prouvé à partir de la définition 13 qui énonce que $\mathsf{FM}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$.

(P2) : par la définition 13, si $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \neq \emptyset$, $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$ conserve uniquement les modèles de forêt qui appartiennent à l'intersection de $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ et $\mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ car $d(\mathcal{F}', \mathcal{F}') = 0$ pour chaque $\mathcal{F}'$ appartenant à cette intersection.

(P3) : par la définition 13, $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$ n'est jamais vide si $\mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ n'est pas vide.

(P4) : c'est une conséquence directe de la définition 13.

(P5) : soit $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}, \mathcal{O}') \cap \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}'))$. Par la définition 13, $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}') \cup \mathsf{sub}(\mathcal{O}'')) \subseteq \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}') \cup \mathsf{sub}(\mathcal{O}''))$, et il existe une $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ qui est la plus proche de $\mathcal{F}'$ car $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}, \mathcal{O}')$.

(P6) : soit $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'')$. Par la définition 13, $\mathcal{F} \in \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}') \cup \mathsf{sub}(\mathcal{O}'))$. Pour montrer que $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathcal{O}')$, on utilise $\mathcal{F}_0 \in \mathsf{FM}(\mathcal{O}, \mathcal{O}') \cap \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}'))$ et le pré-ordre total sur les modèles de forêt.

Nous renvoyons les lecteurs à l'annexe pour une démonstration complète du théorème. □

4.3 Ontologie de révision

Dans cette section, nous présentons une procédure pour construire une ontologie $\mathcal{O}^*$ en $\mathcal{SHIQ}$ qui admet au moins les modèles de forêt dans $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$.

4.3.1 Approximation supérieure

Il a été avéré par [De Giacomo *et al.*, 2007] qu'il peut ne pas exister une ontologie en DL-Lite qui admette exactement un ensemble donné de modèles. Par l'exemple 16, nous montrons que c'est aussi le cas pour des ontologies en $\mathcal{SHIQ}$.

Exemple 16. Reconsidérons l'exemple 15 avec $\mathsf{FM}(\mathcal{O}, \mathcal{O}') = \{\mathcal{F}'_1\}$. Supposons qu'il existe $\widehat{\mathcal{O}}$ avec $\mathsf{sub}(\widehat{\mathcal{O}}) = \{\mathsf{Professor}, \neg\mathsf{Professor}, \mathsf{Researcher}, \neg\mathsf{Researcher}, \mathsf{Expert}, \neg\mathsf{Expert}, \mathsf{Student}, \neg\mathsf{Student}, \mathsf{Course}, \neg\mathsf{Course}, \exists\mathsf{supervises}.\mathsf{Student}, \forall\mathsf{supervises}.(\neg\mathsf{Student}), (\geq 2\ \mathsf{teaches}.\mathsf{Course}), (\leq 1\ \mathsf{teaches}.\mathsf{Course})\}$ qui admette la seule $\mathcal{F}'_1$ en tant que modèle de forêt. En appliquant le nouvel algorithme de tableau sur $\widehat{\mathcal{O}}$, $\mathsf{FM}(\widehat{\mathcal{O}})$ doit contenir $\mathcal{F}'_1$ et une autre $\mathcal{F}'$ ayant un nœud $\{x\}$ avec $L(x) = \{\{\mathsf{Alex}\}, \mathsf{Professor}, \mathsf{Researcher}, \mathsf{Expert}, \neg\mathsf{Student}, \neg\mathsf{Course}, \forall\mathsf{supervises}.(\neg\mathsf{Student}), (\leq 1\ \mathsf{teaches}.\mathsf{Course})\}$, ce qui est une contradiction. Donc, il peut ne pas exister une ontologie en $\mathcal{SHIQ}$ qui admette exactement un ensemble donné de modèles de forêt.

Pour résoudre ce problème, nous empruntons la notion d'*approximation maximale* du travail de De Giacomo *et al.* [De Giacomo *et al.*, 2007] pour définir une *ontologie*

d'approximation supérieure dans notre contexte comme suit.

Définition 14 (**Approximation supérieure**). *Soient $\mathcal{O}$ et $\mathcal{O}'$ deux ontologies cohérentes en $\mathcal{SHIQ}$ avec l'opération de révision $\mathsf{FM}(\mathcal{O},\mathcal{O}')$, nous utilisons $\mathcal{S}(\mathcal{O}'')$ pour désigner la signature d'une ontologie $\mathcal{O}''$. Une ontologie $\mathcal{O}^*$ est une* approximation supérieure *de $\mathsf{FM}(\mathcal{O},\mathcal{O}')$ si :*

1. $\mathcal{S}(\mathcal{O}^*) \subseteq \mathcal{S}(\mathcal{O}) \cup \mathcal{S}(\mathcal{O}')$;
2. $\mathsf{FM}(\mathcal{O},\mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O}^*)$;
3. *il n'existe pas d'ontologie $\mathcal{O}''$ telle que* $\mathsf{FM}(\mathcal{O},\mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O}'') \subset \mathsf{FM}(\mathcal{O}^*)$.

La définition 14 fournit une meilleure ontologie d'approximation à construire de telle sorte qu'elle admette tous les modèles de forêt dans $\mathsf{FM}(\mathcal{O},\mathcal{O}')$. Un point intéressant est que si une telle approximation supérieure existe, elle est unique à l'équivalence sémantique. C'est-à-dire que s'il existe deux approximations supérieures alors elles sont sémantiquement équivalentes. Ceci est confirmé par le lemme suivant.

Lemme 7. *Soient $\mathcal{O}$ et $\mathcal{O}'$ deux ontologies cohérentes en $\mathcal{SHIQ}$ avec l'opération de révision $\mathsf{FM}(\mathcal{O},\mathcal{O}')$. S'il y a deux approximations supérieures $\mathcal{O}^*$ et $\widehat{\mathcal{O}}$ de $\mathsf{FM}(\mathcal{O},\mathcal{O}')$ alors $\mathsf{FM}(\widehat{\mathcal{O}},\mathsf{sub}(\mathcal{O}^*)) = \mathsf{FM}(\mathcal{O}^*,\mathsf{sub}(\widehat{\mathcal{O}}))$.*

Démonstration. Supposons que $\mathsf{FM}(\widehat{\mathcal{O}},\mathsf{sub}(\mathcal{O}^*)) \neq \mathsf{FM}(\mathcal{O}^*,\mathsf{sub}(\widehat{\mathcal{O}}))$. Par la définition 14, nous avons $\mathsf{FM}(\mathcal{O},\mathcal{O}') \subseteq \mathsf{FM}(\widehat{\mathcal{O}})$ et $\mathsf{FM}(\mathcal{O},\mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O}^*)$. Donc, $\mathsf{FM}(\mathcal{O},\mathcal{O}') \subseteq \mathsf{FM}(\widehat{\mathcal{O}}) \cap \mathsf{FM}(\mathcal{O}^*)$ (i).

Cela implique que $\mathsf{FM}(\widehat{\mathcal{O}}) \cap \mathsf{FM}(\mathcal{O}^*) \neq \emptyset$, et donc $\mathsf{sub}(\widehat{\mathcal{O}}) = \mathsf{sub}(\mathcal{O}^*)$ en raison de l'algorithme de tableau avec la sat-règle. Nous obtenons $\mathsf{FM}(\widehat{\mathcal{O}}) = \mathsf{FM}(\widehat{\mathcal{O}},\mathsf{sub}(\mathcal{O}^*))$, $\mathsf{FM}(\mathcal{O}^*) = \mathsf{FM}(\mathcal{O}^*,\mathsf{sub}(\widehat{\mathcal{O}}))$, et donc $\mathsf{FM}(\widehat{\mathcal{O}}) \neq \mathsf{FM}(\mathcal{O}^*)$ (*).

Selon le corollaire 2, nous avons $\mathsf{FM}(\widehat{\mathcal{O}} \cup \mathcal{O}^*) = \mathsf{FM}(\widehat{\mathcal{O}},\mathsf{sub}(\mathcal{O}^*)) \cap \mathsf{FM}(\mathcal{O}^*,\mathsf{sub}(\widehat{\mathcal{O}}))$ lequel permet $\mathsf{FM}(\widehat{\mathcal{O}} \cup \mathcal{O}^*) = \mathsf{FM}(\widehat{\mathcal{O}}) \cap \mathsf{FM}(\mathcal{O}^*)$. Par (*), nous avons $\mathsf{FM}(\widehat{\mathcal{O}}) \cap \mathsf{FM}(\mathcal{O}^*) \subset \mathsf{FM}(\mathcal{O}^*)$ ou $\mathsf{FM}(\widehat{\mathcal{O}}) \cap \mathsf{FM}(\mathcal{O}^*) \subset \mathsf{FM}(\widehat{\mathcal{O}})$ (ii).

(i) et (ii) implique que $\mathsf{FM}(\mathcal{O},\mathcal{O}') \subseteq \mathsf{FM}(\widehat{\mathcal{O}} \cup \mathcal{O}^*) = \mathsf{FM}(\widehat{\mathcal{O}}) \cap \mathsf{FM}(\mathcal{O}^*) \subset X$ pour $X = \mathsf{FM}(\widehat{\mathcal{O}})$ ou $X = \mathsf{FM}(\mathcal{O}^*)$,ce qui contredit la condition 3 dans la définition 14. □ □

Dans la suite de cette section, nous montrerons que l'approximation supérieure de la définition 14 existe réellement et proposent une procédure pour la construire.

4.3.2 Construction de l'ontologie de révision

Définition 15 (**Ontologie de révision**). *Soient* $\mathcal{O} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ *et* $\mathcal{O}' = (\mathcal{T}', \mathcal{R}', \mathcal{A}')$ *deux ontologies cohérentes en* $\mathcal{SHIQ}$ *avec* $\mathsf{FM}(\mathcal{O}, \mathcal{O}') = \{\mathcal{F}_1, \cdots, \mathcal{F}_n\}$ *pour* $1 \leq i \leq n$ *; pour chaque* $\mathcal{F}_i$=$(G_i, T\langle\widehat{x}_1\rangle, \ldots, T\langle\widehat{x}_m\rangle)$ *avec* $G_i = (\mathbf{V}_i, \mathbf{E}_i, \mathbf{L}_i)$ *et* $T\langle\widehat{x}_j\rangle = \langle V_j, L_j, E_j\rangle$ *(*$1 \leq j \leq m$*), soit* $\mathcal{V}_i = \mathbf{V}_i \cup V_1 \cup \cdots \cup V_m$*. Une ontologie de révision* $\mathcal{O}^* = (\widehat{\mathcal{T}}, \widehat{\mathcal{R}}, \widehat{\mathcal{A}})$ *de* $\mathcal{O}$ *par* $\mathcal{O}'$ *est définie comme suit :*

— $\widehat{\mathcal{R}} := \mathcal{R}'$

— $\widehat{\mathcal{T}} := \mathcal{T}' \cup \{\top \sqsubseteq \bigsqcup_{1\leq i\leq n} (\bigsqcup_{x\in\mathcal{V}_i} (\bigsqcap_{C\in L_i(x)} C))\}$

— $\widehat{\mathcal{A}}$ *contient un ensemble d'assertions :*

$\{C(x) \in \mathcal{A}'\} \cup \{R(x,y) \in \mathcal{A}'\} \ \cup$
$\{C(x) \in \mathcal{A} \mid X \subseteq L_i(x), X \in \mathsf{Flat}(C), 1 \leq i \leq n\} \ \cup$
$\{R(x,y) \in \mathcal{A} \mid R \in L_i(\langle x, y\rangle), 1 \leq i \leq n\} \ \cup$
$\{x \not\doteq y \mid x, y \in \mathbf{I} \cup \mathbf{I}', x \not\doteq y \in \mathcal{F}_i, 1 \leq i \leq n\}$

La construction d'une ontologie de révision $\mathcal{O}^*$ conserve tous les axiomes de concept et de rôle ainsi que les assertions de $\mathcal{O}'$. Elle ajoute aussi dans $\mathcal{O}^*$ un nouvel axiome de concept construit littéralement depuis $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$. Lors de la construction d'un modèle de forêt $\mathcal{F}$ en appliquant l'algorithme de tableau sur $\mathcal{O}^*$, cet axiome de concept force à choisir un nœud x d'un modèle de forêt $\mathcal{F}_i \in \mathsf{FM}(\mathcal{O}, \mathcal{O}')$ pour ajouter son étiquette $L(x)$ dans celle du nœud actuel de $\mathcal{F}$. En dehors des modèles de forêt dans $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$, l'algorithme de tableau peut construire un modèle de forêt dont les nœuds possèdent des étiquettes venant d'autres modèles de forêt de $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$. C'est pourquoi $\mathsf{FM}(\mathcal{O}^*)$ peut être plus grand que $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$.

Notons que l'axiome de concept construit à partir de $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$ permet de capturer la partie sémantique des seuls axiomes de concept et de rôle de $\mathcal{O}$ qui devrait être propagés

à $\mathcal{O}^*$. Pour transférer des parties sémantiques des assertions de $\mathcal{O}$ à $\mathcal{O}^*$, il est nécessaire de déterminer les assertions de $\mathcal{O}$ qui restent à satisfaire dans les modèles de forêt de $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$. Cela signifie qu'il peut y avoir une assertion de $\mathcal{O}$ qui ne peut pas être propagée à $\mathcal{O}^*$. Par exemple, $\mathcal{O}$ contient les assertions $R(a, b)$, $S(a, c)$ alors que $\mathcal{O}'$ contient les axiomes $\top \sqsubseteq \forall R.\bot$ et $\top \sqsubseteq \forall S.\top$. Par construction, chaque modèle de forêt pour $\mathcal{O}'$ possède une arête vide entre a et b mais a est un S-voisin de c. Cela implique que $S(a, c)$ sera ajouté dans $\mathcal{O}^*$ mais $R(a, b)$ ne le sera pas.

Exemple 17. Continuons l'exemple 15, construisons depuis $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$ une ontologie $\mathcal{O}^*$ qui admette un seul modèle de forêt $\mathcal{F}_1'$ d'après la définition 15. Donc, $\mathcal{O}^*$ contient les axiomes suivants :

$\mathsf{Expert} \sqsubseteq \forall \mathsf{supervises}.(\neg \mathsf{Student})$, $\mathsf{Researcher} \sqsubseteq \forall \mathsf{supervises}.(\neg \mathsf{Student})$ (de $\mathcal{O}'$),

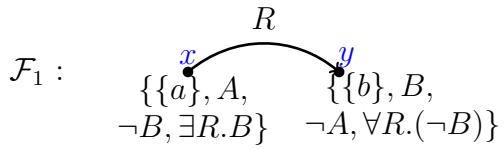
et

$\top \sqsubseteq (\mathsf{Professor} \sqcap \neg \mathsf{Researcher} \sqcap \mathsf{Expert} \sqcap \neg \mathsf{Course} \sqcap \neg \mathsf{Student} \sqcap \exists \mathsf{supervises}.\mathsf{Student} \sqcap (\geq 2\ \mathsf{teaches}.\mathsf{Course})) \sqcup (\mathsf{Professor} \sqcap \mathsf{Researcher} \sqcap \neg \mathsf{Expert} \sqcap \neg \mathsf{Course} \sqcap \neg \mathsf{Student} \sqcap \exists \mathsf{supervises}.\mathsf{Student} \sqcap (\geq 2\ \mathsf{teaches}.\mathsf{Course})) \sqcup (\mathsf{Student} \sqcap \forall \mathsf{supervises}.(\neg \mathsf{Student}) \sqcap (\leq 1\ \mathsf{teaches}.\mathsf{Course}) \sqcap \neg \mathsf{Course} \sqcap \neg \mathsf{Professor} \sqcap \neg \mathsf{Researcher} \sqcap \neg \mathsf{Expert}) \sqcup (\mathsf{Course} \sqcap \forall \mathsf{supervises}.(\neg \mathsf{Student}) \sqcap (\leq 1\ \mathsf{teaches}.\mathsf{Course}) \sqcap \neg \mathsf{Student} \sqcap \neg \mathsf{Professor} \sqcap \neg \mathsf{Researcher} \sqcap \neg \mathsf{Expert})$, $\mathsf{Professor}(\mathsf{Alex})$.

L'exemple 18 illustre également tout un processus de révision d'ontologie dans lequel les parties sémantiques d'une ontologie initiale $\mathcal{O}$ sont transférées à $\mathcal{O}^*$.

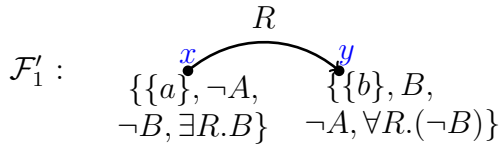
Exemple 18. Étant donné une ontologie $\mathcal{O}$ constituée des axiomes et assertions suivants : $A \sqsubseteq \exists R.B, B \sqsubseteq \neg A \sqcap \forall R.(\neg B), A(a), B(b), R(a, b)$. Soit $\mathcal{O}'$ une ontologie consistant en la seule assertion $\neg A(a)$. Notons qu'il n'y a aucun rôle se trouvant dans $\mathcal{O}'$. Il s'ensuit que la $\mathsf{sat}_{\mathcal{R}}$-règle n'est pas appliquée lors de l'application de l'algorithme de tableau sur $\mathcal{O}$ pour construire l'ensemble $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$. Donc, nous obtenons $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ contenant un seul modèle de forêt $\mathcal{F}_1$ comme suit :

$\mathcal{F}_1$:

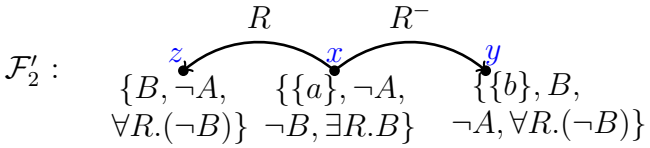


En appliquant l'algorithme de tableau sur $\mathcal{O}'$ pour construire l'ensemble $\mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ avec la $\mathsf{sat}_{\mathcal{R}}$-règle fonctionnant sur $\mathsf{sub}_{\mathcal{R}} = \mathsf{sub}_{\mathcal{R}}(\mathcal{O}) = \{R, R^-\}$, nous obtenons au moins 3 modèles de forêt $\mathcal{F}'_1$, $\mathcal{F}'_2$ et $\mathcal{F}'_3$ respectivement comme suit :

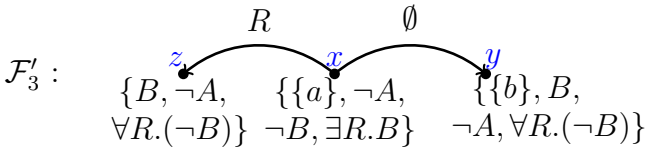
$\mathcal{F}'_1$:



$\mathcal{F}'_2$:



$\mathcal{F}'_3$:



D'autres modèles de forêt dans $\mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ sont des variantes des trois forêts ci-dessus en changeant les concepts dans chaque étiquette de nœud. Ensuite, en calculant la distance entre chaque modèle de forêt dans $\mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ avec l'une d'elles dans $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$, nous obtenons $d(\mathcal{F}'_1, \mathcal{F}_1) = 2$ qui est minimale. Donc, par la définition 13, $\mathsf{FM}(\mathcal{O}, \mathcal{O}') = \{\mathcal{F}'_1\}$. Enfin, l'ontologie de révision $\mathcal{O}^*$ peut être construite selon la définition 15 pour obtenir les axiomes et assertions $\top \sqsubseteq (\neg A \sqcap \neg B \sqcap \exists R.B) \sqcup (B \sqcap \neg A \sqcap \forall R.(\neg B))$, $\neg A(a), B(b), R(a, b)$.

Formulons et prouvons maintenant un résultat important affirmant que l'ontologie de révision $\mathcal{O}^*$ définie pour deux ontologies $\mathcal{O}$ et $\mathcal{O}'$ selon la définition 15 est une approximation supérieure de $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$. Notre argument s'appuie fortement sur le comportement spécifique de la sat-règle et la particularité de l'axiome de concept ajouté à $\mathcal{O}^*$. Effectivement, connaissant le résultat de l'application de la sat-règle (*i.e.* le sous-ensemble S choisi à partir de $\mathsf{sub}(\mathcal{O})$) à chaque nœud d'un modèle de forêt, on connaît aussi tout le modèle de forêt.

Théorème 5. *Soient $\mathcal{O}$ et $\mathcal{O}'$ deux ontologies cohérentes en $\mathcal{SHIQ}$. L'ontologie de révision $\mathcal{O}^*$ de $\mathcal{O}$ par $\mathcal{O}'$ est une approximation supérieure de $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$. De plus, la taille de $\mathcal{O}^*$ est bornée par une fonction doublement exponentielle par rapport à la taille de $\mathcal{O}$ et $\mathcal{O}'$.*

Résumé de démonstration. Afin de démontrer le théorème, nous devons montrer que les conditions de la définition 14 sont satisfaites. La première condition est simple par construction. Une preuve de la deuxième condition peut être fondée sur les particularités de la sat-règle dans l'algorithme de tableau et la structure de $\mathcal{O}^*$. Soit $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathcal{O}')$, si nous exécutons l'algorithme de tableau sur $\mathcal{O}^*$, il peut commencer la construction d'une forêt en appliquant les sat-règle et $\mathsf{sat}_{\mathcal{R}}$-règle à des nœuds racines et des arêtes de racines de telle sorte qu'elles obtiennent les mêmes étiquettes à partir des nœuds racines et des arêtes de racines correspondantes dans $\mathcal{F}$. Pour chaque nœud non-racine x de $\mathcal{F}$, la sat-règle peut choisir un certain $S \subseteq \mathsf{sub}(\mathcal{O}^*)$ tel que $S = L(x)$. Cela signifie que l'on peut utiliser un modèle de forêt $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathcal{O}')$ pour "guider" l'algorithme de tableau lors de la construction de le même modèle de forêt sur $\mathcal{O}^*$. Concernant la troisième condition, nous pouvons utiliser le raisonnement par l'absurde qui permet de déduire $\mathsf{FM}(\mathcal{O}'') = \mathsf{FM}(\mathcal{O}^*)$ de l'existence d'une telle ontologie $\mathcal{O}''$. Pour ce faire, nous établissons d'abord $\mathsf{sub}(\mathcal{O}'') = \mathsf{sub}(\mathcal{O}^*)$. Ensuite, nous montrons que l'existence d'une séquence d'applications de règles (incluant la sat-règle) sur $\mathcal{O}^*$ donnant un modèle de forêt $\mathcal{F} \in \mathsf{FM}(\mathcal{O}^*)$ implique l'existence d'une séquence d'applications de règles sur $\mathcal{O}''$ donnant un modèle de forêt $\mathcal{F}'' \in \mathsf{FM}(\mathcal{O}'')$. Vous trouverez une preuve complète en annexe. □

4.4 Conclusion

Nous avons présenté dans ce chapitre une approche sémantique pour la révision d'une ontologie en $\mathcal{SHIQ}$ avec individus. Une caractéristique intéressante de notre approche est d'introduire des structures finies, à savoir les *forêts de complétion*, pour caractériser la sémantique d'une ontologie en $\mathcal{SHIQ}$. Ces structures peuvent être construites par un

algorithme de tableau explorant tout non-déterminisme intrinsèque découlant des disjonctions et des restrictions de cardinalité dans une ontologie. La distance sémantique entre des ontologies expressives peut maintenant être traduite sur une distance entre des forêts de complétion. Cette fonctionnalité est cruciale pour définir une opération de révision qui assure le changement minimal. Grâce à cette distance, nous sommes en mesure de déterminer un ensemble de forêts de complétion que l'ontologie révisée devrait admettre. Pour résoudre le problème de l'inexpressivité, *i.e.* la non-existence d'une ontologie en $\mathcal{SHIQ}$ qui admette exactement un ensemble de forêts de complétion comme des modèles, nous avons introduit la notion d'ontologie par approximation supérieure. Cela nous a amené à étendre l'ensemble des forêts de complétion pour caractériser la sémantique d'une ontologie en ajoutant une nouvelle règle d'expansion, à savoir sat-règle, à notre algorithme de tableau. Bien que le comportement non déterministe de cette règle trahisse de bonnes caractéristiques des algorithmes de tableau standards, cet inconvénient est justifié par l'obtention d'une opération de révision qui garantit tous les postulats de révision (dont le changement minimal) et une ontologie révisée exprimable en $\mathcal{SHIQ}$.

Troisième partie

Mise en œuvre

Chapitre 5

Techniques d'optimisation

Dans la partie précédente, nous avons prouvé la décidabilité de la révision d'ontologies en $\mathcal{SHIQ}$ en proposant une procédure permettant de construire une ontologie révisée qui prend en compte les nouveaux axiomes et reste sémantiquement la plus proche de l'ontologie initiale. Nous avons également montré que la taille de l'ontologie révisée est bornée par une fonction triplement exponentielle de la taille de l'ontologie initiale. Cette importante complexité n'est pas surprenante et provient principalement des sources suivantes : (i) la caractérisation de la sémantique d'une ontologie en utilisant des modèles de forêt obtenus de l'exploration de toutes les branches non déterministes, *i.e.* la taille de $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$, peut atteindre une fonction triplement exponentielle de la taille des ontologies $\mathcal{O}$ et $\mathcal{O}'$; (ii) le calcul de la distance entre deux modèles de forêt peut être exponentiel en fonction de la taille des modèles de forêt, et (iii) la construction d'une ontologie d'approximation supérieure oblige l'algorithme tableau à utiliser la sat-règle qui considère des cas exhaustivement non déterministes.

Dans ce chapitre, nous présentons des techniques d'optimisation pour réduire la complexité découlant des sources mentionnées. Ces techniques s'appliquant aux forêts de complétion (*cf.* chapitre 4) peuvent également s'appliquer aux arbres de complétion (*cf.* chapitre 3). Nous commencerons ici par présenter une technique d'*absorption* qui est développée pour réduire des non-déterminismes lors de l'application de l'algorithme de tableau.

5.1 Absorption

Comme mentionné dans les parties précédentes, une ontologie $\mathcal{O}$ inclut un ensemble d'axiomes GCI $\{C_i \sqsubseteq D_i\}$ où C_i, D_i peuvent être complexes. Notre algorithme de tableau s'appliquant à $\mathcal{O}$ peut construire un graphe qui représente un modèle de $\mathcal{O}$. Afin de satisfaire chaque axiome GCI $C_i \sqsubseteq D_i$ (*i.e.* $(C_i)^{\mathcal{I}} \subseteq (D_i)^{\mathcal{I}}$ pour un modèle $\mathcal{I}$), l'algorithme de tableau doit ajouter à l'étiquette de chaque nœud x un sous-ensemble de concepts $X \in \mathsf{Flat}(\neg C_i \sqcup D_i)$ selon un comportement de la sat-règle. Ce comportement vient du fait que $C_i \sqsubseteq D_i$ est équivalent à $\top \sqsubseteq C_i \sqcup D_i$. Par conséquent, la satisfaction d'un axiome GCI $C_i \sqsubseteq D_i$ qui est initialement déterministe entraîne un indéterminisme. Notons qu'une des raisons de la complexité exponentielle de l'algorithme de tableau provient de cet indéterminisme dont la nuisance est de nature globale car il doit être traité pour chaque nœud du graphe.

Pour éviter de traiter de tels cas non-déterministes, il est donc logique d'éliminer des axiomes GCI de l'ontologie chaque fois que possible. L'absorption est une technique qui tente de le faire en réécrivant chaque $C_i \sqsubseteq D_i$ en un axiome $A \sqsubseteq D'_i$ où A est un nom de concept [Horrocks, 2003]. Dans ce cas, on dit que A est absorbé. La technique d'absorption stipule que si A est atomique (et $\neg A$ n'est pas absorbé) alors il suffit de choisir un sous-ensemble $Y \in \mathsf{Flat}(D'_i)$ afin de l'ajouter dans $L(x)$ si $A \in L(x)$. Cette technique fonctionne car si $A \notin L(x)$ alors $L(x)$ peut être complétée avec $\neg A$. Cette complétion n'entraîne pas de déclenchement d'autre règle d'expansion car A est atomique et $\neg A$ n'est pas absorbé. Donc, l'axiome $\top \sqsubseteq A \sqcup D'_i$ est vérifié.

La technique d'absorption de base a été raffinée et étendue de plusieurs façons :

— L'*absorption de rôle* [Tsarkov and Horrocks, 2004] réécrit chaque GCI à la forme de $\exists R.A \sqsubseteq D$ où A est un concept atomique et $\neg A$ n'est pas absorbé. Dans ce cas, on surveille chaque changement d'étiquette de chaque nœud ou la création d'un nœud. Si $\exists R.A \in L(x)$ est trouvé, un sous-ensemble $Y \in \mathsf{Flat}(D)$ est choisit pour être ajouté dans $L(x)$. De plus, s'il a un R-voisin y de x avec $A \in L(y)$ alors on ajoute $\exists R.A$ et Y dans $L(x)$. Cette absorption fonctionne car $\forall R.(\neg A)$ peut être ajouté dans $L(x)$ si $\exists R.A \notin L(x)$. Cela peut entraîner l'ajout de $\neg A$ dans chaque R-voisin y de x.

Puisque $\neg A$ n'est pas absorbé, aucune autre application de règle ne sera déclenchée.

— L'*absorption binaire* [Hudek and Weddell, 2006] réécrit chaque GCI à la forme de $A \sqcap B \sqsubseteq D$ où A, B sont des concepts atomiques et $\neg A, \neg B$ ne sont pas absorbés. Dans ce cas, on surveille chaque changement d'étiquette de chaque nœud. Si l'on trouve $\{A, B\} \subseteq L(x)$, un sous-ensemble $Y \in \mathsf{Flat}(D)$ est choisi pour être ajouté dans $L(x)$. Cette absorption fonctionne car (i) si $A \notin L(x)$ (ou $B \notin L(x)$) on peut ajouter $\neg A$ (ou $\neg B$) dans $L(x)$ sans déclencher d'autres applications de règle car $\neg A$ (ou $\neg B$) n'est pas absorbé.

En résumé, l'idée clé sous-jacente à l'application de l'absorption est la possibilité de gérer (c'est-à-dire surveiller) les changements du graphe pour que toute satisfaction (implicite) du subsumé de certains axiomes sur un nœud soit détectée.

5.2 Calcul de la distance entre deux modèles de forêt

Selon la formule de la distance entre deux modèles de forêt (*cf.* Définition 12), il existe un unique isomorphisme entre deux nœuds de racine de deux forêts modèle $\mathcal{F}$=$(G, T\langle\widehat{x}_1\rangle, \ldots, T\langle\widehat{x}_m\rangle)$ avec $G = (\mathbf{V}, \mathbf{E}, \mathbf{L})$ et $T\langle\widehat{x}_i\rangle = \langle V_i, L_i, E_i\rangle$ $(1 \leq i \leq m)$, et $\mathcal{F}'$=$(G', T\langle\widehat{x}'_1\rangle, \ldots, T\langle\widehat{x}'_m\rangle)$ avec $G' = (\mathbf{V}', \mathbf{E}', \mathbf{L}')$ et $T\langle\widehat{x}'_j\rangle = \langle V'_j, L'_j, E'_j\rangle$ $(1 \leq j \leq m)$. Donc, il suffit d'étudier l'optimisation du calcul de la distance entre deux structures d'arbres $T\langle\widehat{x}_i\rangle$ et $T\langle\widehat{x}'_j\rangle$.

Soient $T\langle x_0\rangle = (V_1, E_1, L_1)$, $T\langle z_0\rangle = (V_2, E_2, L_2)$ deux arbres. Par la définition 12, nous avons

$$\begin{aligned} d(T\langle x_0\rangle, T\langle z_0\rangle) = \min_{\pi \in \Pi(T\langle x_0\rangle, T\langle z_0\rangle)} \{ \max_{\langle x,y\rangle \in E_1} (&|L_1(x) \vartriangle L_2(\pi(x))| \\ &+ |L_1(\langle x, y\rangle) \vartriangle L_2(\langle \pi(x), \pi(y)\rangle)| \\ &+ |L_1(y) \vartriangle L_2(\pi(y))|)\} \end{aligned}$$

où $\Pi(T\langle x_0\rangle, T\langle z_0\rangle)$ est l'ensemble de tous les isomorphismes de $T\langle x_0\rangle$ à $T\langle z_0\rangle$. Par la suite, nous présentons un algorithme pour calculer $d(T\langle x_0\rangle, T\langle z_0\rangle)$ fonctionnant en temps

polynomial en taille de $T\langle x_0\rangle$ et $T\langle z_0\rangle$. Les idées principales de l'algorithme sont fondées sur les observations suivantes :

— Étant donné un isomorphisme π, nous désignons
$h(\pi) = \max_{\langle x,y\rangle \in E_1}(|L_1(x) \vartriangle L_2(\pi(x))| + |L_1(\langle x, y\rangle) \vartriangle L_2(\langle \pi(x), \pi(y)\rangle)| + |L_1(y) \vartriangle L_2(\pi(y))|)$
Il y a au plus $O(\ell)$ différentes de $h(\pi)$ où ℓ est la taille maximale de $\mathcal{O}$ et $\mathcal{O}'$. En effet, par construction, nous avons $|L(x)| \leq O(\ell)$ et $|L(\langle x, y\rangle)| \leq O(\ell)$ pour chaque nœud x et arête $\langle x, y\rangle$ des arbres. Cela nous permet de partitionner $\Pi(T\langle x_0\rangle, T\langle z_0\rangle)$ en plusieurs groupes dont chacun correspond à une valeur $v_i \in \Delta$ où $v_{i-1} > v_i$ pour tout $2 \leq i \leq m$.

— Pour chaque valeur $v_i \in \Delta$ de la plus grande à la plus petite, il est possible de déterminer de façon polynomiale s'il existe un isomorphisme $\pi \in \Pi(T\langle x_0\rangle, T\langle z_0\rangle)$ tel que $v_i > h(\pi)$. S'il n'existe pas un tel isomorphisme π, nous obtenons $d(T\langle x_0\rangle, T\langle z_0\rangle) = v_i$; sinon, l'algorithme considère la valeur v_{i+1}.

Si deux arbres $T\langle x_0\rangle$ et $T\langle z_0\rangle$ ont uniquement des nœuds racines (x_0 et z_0) alors

$$d(T\langle x_0\rangle, T\langle z_0\rangle) = |L_1(x_0) \vartriangle L_2(z_0)|$$

Sinon, on peut toujours obtenir deux arbres ayant la même structure en y ajoutant des nœuds et des arêtes vides, *e.g.* les deux arbres de complétion présentés dans la figure 5.1. Deux arêtes $\langle x_i, x_j\rangle \in T\langle x_0\rangle$ et $\langle z_{i'}, z_{j'}\rangle \in T\langle z_0\rangle$ forment un lien s'ils sont situés au même niveau dans les deux arbres. Un lien $(\langle x_i, x_j\rangle, \langle z_{i'}, z_{j'}\rangle)$ est à un niveau i si x_j et $z_{j'}$ sont au niveau i. Par exemple, $(\langle x_1, x_3\rangle, \langle z_1, z_4\rangle)$ est un lien au niveau 2 de la figure 5.1 mais $(\langle x_1, x_3\rangle, \langle z_0, z_1\rangle)$ n'est pas un lien. De plus, nous définissons un ensemble $\mathsf{LINKS}(\mathsf{succ}(x), \mathsf{succ}(z)) = \{(\langle x, x_i\rangle, \langle z, \phi(x_i)\rangle) \mid \phi$ est une bijection de $\mathsf{succ}(x)$ à $\mathsf{succ}(z)\}$ où $\mathsf{succ}(x)$ désigne l'ensemble des successeurs de x.

Nous pouvons construire polynomialement un ensemble qui contient des valeurs $d(\langle x_i, x_j\rangle, \langle z_{i'}, z_{j'}\rangle)$ définies comme suit : $d(\langle x_i, x_j\rangle, \langle z_{i'}, z_{j'}\rangle) = |L(x_i) \vartriangle L(z_{i'})| + |L(\langle x_i, x_j\rangle) \vartriangle L(\langle z_{i'}, z_{j'}\rangle)| + |L(x_j) \vartriangle L(z_{j'})|$ pour chaque lien $(\langle x_i, x_j\rangle, \langle z_{i'}, z_{j'}\rangle)$ à chaque niveau de deux arbres. En particulier, nous définissons $d(x_0, z_0) = |L(x_0) \vartriangle L(z_0)|$

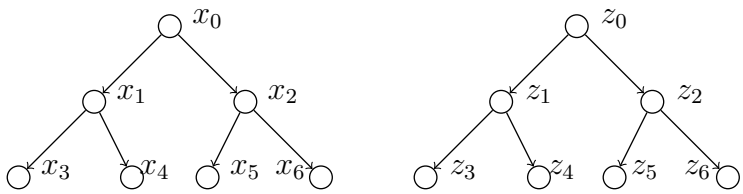


FIGURE 5.1 – Deux arbres $T\langle x_0\rangle$ (à gauche) et $T\langle z_0\rangle$ (à droite)

pour deux nœuds racines x_0 et z_0. Par exemple, au niveau 1 des deux arbres de la figure 5.1, cet ensemble inclut les valeurs suivantes : $d(\langle x_0, x_1\rangle, \langle z_0, z_1\rangle)$, $d(\langle x_0, x_1\rangle, \langle z_0, z_2\rangle)$, $d(\langle x_0, x_2\rangle, \langle z_0, z_1\rangle)$, $d(\langle x_0, x_2\rangle, \langle z_0, z_2\rangle)$.

Soit Δ la liste triée incluant toutes les valeurs de l'ensemble ci-dessus, *i.e.* $\Delta = \{v_1, \ldots, v_m\}$ où $v_k > v_j$ pour $1 \leq k < j \leq m$. Pour chaque $v_k \in \Delta$, nous définissons $\mathsf{COOR}(v_k) = \{(\langle x_i, x_j\rangle, \langle z_{i'}, z_{j'}\rangle) \mid d(\langle x_i, x_j\rangle, \langle z_{i'}, z_{j'}\rangle) = v_k\}$. Par la suite, nous disons qu'un isomorphisme π passe par un lien $(\langle x_i, x_j\rangle, \langle z_{i'}, z_{j'}\rangle)$ si $\pi(x_i) = z_{i'}$ et $\pi(x_j) = z_{j'}$. Étant donné deux arbres de complétion isomorphes, selon la Définition 12, il y a un isomorphisme π et un lien $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$ tels que $d(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$ équivaille à la distance entre ces arbres. Le lemme suivant fournit plus de précision pour cette observation.

Lemme 8. *Une valeur $v_k \in \Delta$ est la distance entre $T\langle x_0\rangle$ et $T\langle z_0\rangle$ si et seulement si,*

(i) il existe un isomorphisme π et un lien $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$ tels que $d(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle) = v_k$ et $d(\langle w, z\rangle, \langle \pi(w), \pi(z)\rangle) \leq v_k$ pour tous les liens $(\langle w, z\rangle, \langle \pi(w), \pi(z)\rangle)$ différents de $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$, et

(ii) pour chaque isomorphisme π, il existe un lien $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$ tel que $d(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle) \geq v_k$.

En se fondant sur le lemme 8, nous proposons l'algorithme 3 pour calculer la distance entre deux arbres. Il cherche à construire un isomorphisme entre deux arbres en "coloriant" des liens dont chacun représente une correspondance entre deux arêtes situées sur deux arbres. L'algorithme 3 prend en entrée deux arbres $T\langle x_0\rangle, T\langle z_0\rangle$ d'hauteur h et renvoie la distance entre $T\langle x_0\rangle$ and $T\langle z_0\rangle$. Cet algorithme 3 procède comme suit. Si $T\langle x_0\rangle$ et $T\langle z_0\rangle$ ont uniquement les nœuds de racine x_0 et z_0, il renvoie $d(x_0, z_0)$ (*cf.* lignes 1 et 2). Sinon, il vérifie si une valeur $v_k \in \Delta$ est la distance entre deux arbres (ligne 4 à ligne 18). Pour ce faire, pour chaque valeur $v_k \in \Delta$ (ligne 4), il colorie en rouge pour les liens dans $\mathsf{COOR}(v)$

pour toute $v \in \Delta$ avec $v \geq v_k$ (ligne 5) et réitère pour fixer la couleur pour d'autres liens comme suit (ligne 6 à ligne 16). Pour chaque lien $(\langle x, y\rangle, \langle z, w\rangle)$ à chaque niveau n de h à 1, il colorie en rouge pour $(\langle x, y\rangle, \langle z, w\rangle)$ si (i) il existe un lien rouge $(\langle x', y'\rangle, \langle z', w'\rangle)$ tel que y' (resp. w') soit un ancêtre de y (resp. w) ou (ii) il n'existe aucun ensemble $\mathsf{LINKS}(\mathsf{succ}(y), \mathsf{succ}(w))$ dont chaque lien est vert (sauf pour le cas où y et w sont des nœuds feuilles et le point (i) n'est pas vrai), alors $(\langle x, y\rangle, \langle z, w\rangle)$ est colorié en vert. Sinon, il colorie en vert pour $(\langle x, y\rangle, \langle z, w\rangle)$. Donc, chaque lien $(\langle x, y\rangle, \langle z, w\rangle)$ sera colorié soit en rouge, soit en vert. Finalement, la valeur v_k est la distance entre deux arbres s'il n'existe aucun ensemble $\mathsf{LINKS}(\mathsf{succ}(x_0), \mathsf{succ}(z_0))$ dont chaque lien est vert (lignes 17 et 18).

Algorithme 3 : computeDistance()

Entrée : $T\langle x_0\rangle, T\langle z_0\rangle$: deux arbres d'hauteur h
Sortie : la distance entre $T\langle x_0\rangle$ et $T\langle z_0\rangle$

```
1  si T⟨x0⟩ et T⟨z0⟩ ont uniquement les nœuds racine x0 et z0 alors
2      Retourner d(x0, z0);
3  Construire COOR(v) pour tout v ∈ Δ ;
4  pour chaque vk ∈ Δ avec 1 ≤ k ≤ m faire
5      Colorier en rouge pour les liens dans COOR(v) pour toute v ∈ Δ avec v ≥ vk;
6      pour chaque n de h à 1 faire
7          pour chaque lien (⟨x, y⟩, ⟨z, w⟩) au niveau n faire
8              si il existe un lien rouge (⟨x', y'⟩, ⟨z', w'⟩) tel que y' (resp. w') est un
                ancêtre de y (resp. w) alors
9                  Colorier en rouge pour (⟨x, y⟩, ⟨z, w⟩) ;
10             sinon si il existe un ensemble LINKS(succ(y), succ(w)) dont chaque lien
                est vert alors
11                 Colorier en vert pour (⟨x, y⟩, ⟨z, w⟩) ;
12             sinon
13                 si (n = h) alors
14                     Colorier en vert pour (⟨x, y⟩, ⟨z, w⟩) ;
15                 sinon
16                     Colorier en rouge pour (⟨x, y⟩, ⟨z, w⟩) ;
17     si il n'existe aucun ensemble LINKS(succ(x0), succ(z0)) dont chaque lien est vert
        alors
18         Retourner vk;
```

Nous formulons et prouvons maintenant le lemme suivant affirmant la correction et la complétude de l'algorithme 3.

Lemme 9. *1. Soient $T\langle x_0\rangle = (V_1, E_1, L_1)$, $T\langle z_0\rangle = (V_2, E_2, L_2)$ deux arbres. L'algorithme 3 retourne la distance entre $T\langle x_0\rangle$ et $T\langle z_0\rangle$.*

2. L'algorithme 3 s'exécute en temps polynomial en fonction de la taille de $T\langle x_0\rangle$ et de $T\langle z_0\rangle$.

Résumé de la démonstration. L'algorithme 3 commence par choisir une valeur v_k de Δ (un ensemble trié de toutes les valeurs de "candidates" pour la distance), prend un lien $(\langle x, y\rangle, \langle z, w\rangle)$ tel qu'il n'ait pas un "mauvais" lien d'ancêtre (un lien $(\langle x, y\rangle, \langle z, w\rangle)$ est "mauvais" si $d(\langle x, y\rangle, \langle z, w\rangle) > v_k$), et puis tente de colorier en vert les liens $\mathsf{LINKS}(\mathsf{succ}(x), \mathsf{succ}(y))$. Le processus de coloriage commence depuis les feuilles vers les nœuds racine. Si ce processus se termine et réussit à colorier les liens racines en vert alor un isomorphisme passant par les liens verts $(\langle x, y\rangle, \langle z, w\rangle)$ peut être construit tel que $d(\langle x, y\rangle, \langle z, w\rangle) \leq v_k$. Cela implique que v_k n'est pas une distance, et l'algorithme choisit une plus petite valeur v_{k-1} de Δ pour la vérifier. Sinon, l'algorithme 3 doit retourner v_k comme une distance entre $T\langle x_0\rangle$ et $T\langle z_0\rangle$. La traçabilité de l'algorithme 3 est une conséquence de celle du processus de coloriage. Une preuve complète peut être trouvée en Annexe. □

5.3 Construction de $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$

Selon la définition 13, $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$ est construit à partir de $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ et $\mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ où $\mathcal{O}$ est beaucoup plus grand que $\mathcal{O}'$. Comme décrit dans le chapitre 4, l'algorithme de tableau doit trouver toutes les forêts de complétion de $\mathcal{O}$ pour construire $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$. Cette construction implique au moins deux sources de complexité : (i) l'explosion exponentielle résultant de des opérateurs de disjonction et de restriction de cardinalité qui se trouvent dans $\mathcal{O}$, et (ii) l'explosion exponentielle découlant du comportement de la sat-règle.

Pour répondre à la première source de complexité, nous utilisons différentes techniques d'optimisation dans la littérature comme des absorptions basiques et binaires [Horrocks, 2003; Tsarkov and Horrocks, 2004; Hudek and Weddell, 2006]. Cependant, cette complexité

appartient intrinsèquement à notre caractérisation de la sémantique d'ontologies puisque nous avons besoin d'un modèle pour chaque cas non-déterministe intrinsèque. Ainsi, la construction de $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ est aussi complexe que de répondre "oui" à une requête telle que $\mathcal{O} \models C \sqsubseteq D$ car un raisonneur doit considérer tous les cas non-déterministes.

Pour aborder la deuxième source de complexité, nous effectuons la construction $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$ en plusieurs étapes :

1. construire $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ sans la sat-règle. Cette étape doit explorer tous les cas non-déterministes intrinsèques découlant de la disjonction et des restrictions de cardinalité. La complexité de cette étape est comparable à celle des algorithmes de tableau standard. Pour réduire le non-déterminisme non intrinsèque issu de la $\sqsubseteq$-règle, nous avons utilisé différentes techniques d'absorption présentées ;

2. appliquer d'une façon indirecte la sat-règle en utilisant des techniques d'absorption pour saturer les étiquettes de nœuds et d'arêtes dans des forêts de complétion de $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$. Cela nous permet d'éviter de construire explicitement des forêts de complétion qui peuvent être déduites des autres forêts de complétion. Par exemple, s'il y a une forêt $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ qui inclut un nœud x tel que $A \in L(x)$ et $\neg A$ ne soit pas absorbé (A est un nom de concept) alors il y a une autre forêt $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ qui peut être obtenue à partir de $\mathcal{F}$ en remplaçant A dans $L(x)$ avec $\neg A$. Cette observation peut être étendue aux concepts de la forme $\exists R.A \in L(x)$;

3. construire $\mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ en propageant des étiquettes de nœuds et d'arêtes des forêts de complétion dans $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$. Cette construction peut être guidée en estimant à la volée la distance entre les forêts de complétion $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ et une nouvelle forêt de complétion $\mathcal{F}'$ construite pour $\mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$. Par exemple, si une estimation donne $d(\mathcal{F}', \mathcal{F}_1) > d(\mathcal{F}'', \mathcal{F}_2)$ avec $\mathcal{F}'' \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ et $\mathcal{F}_1, \mathcal{F}_2 \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ alors il n'est pas nécessaire de terminer la construction de $\mathcal{F}'$.

   Plus précisément, la propagation peut être réalisée en maintenant une fonction qui associe un nœud d'une forêt de complétion $\mathcal{F}'$ en cours de construction à un nœud d'une forêt de complétion $\mathcal{F}$ dans $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$. Cette fonction peut associer deux nœuds racine initialisés à partir d'un même individu ou de deux nœuds (au même niveau) générés à partir d'une même règle d'expansion. En utilisant des associations

entre nœuds de $\mathcal{F}'$ et $\mathcal{F}$ ainsi que des techniques d'absorption, au lieu d'appliquer directement la sat-règle à $\mathcal{F}'$, pour chaque nœud x' de $\mathcal{F}'$, nous pouvons saturer $L(x')$ en ajoutant dans $L(x')$ un sous-ensemble $S \subseteq L(x)$ tel que S est compatible (non-clash) avec $L(x')$ où x est un nœud de $\mathcal{F}$ qui est associé avec x'. Ce comportement peut être meilleur que ceux de la sat-règle car le choix d'un tel sous-ensemble S se fait de $L(x)$ qui est beaucoup plus petit que $\mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$;

4. choisir des forêts de complétion de $\mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ pour construire $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$ en utilisant l'Algorithme 3 pour calculer la distance entre des forêts de complétion de $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ et $\mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$.

5.4 Autres techniques d'optimisation

Contrairement à l'approche précédente de la construction $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$, une autre idée pouvant aussi aider à réduire le nombre de modèles de forêt construits pour calculer $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$ provient de la définition 13 (opération de révision). Dans ce cas, nous construisons d'abord l'ensemble $\mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ en appliquant l'algorithme de tableau. Ensuite, pour chaque modèle de forêt $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$, si l'on retire un "bon" candidat $\mathcal{F}$ depuis $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ tel que $d(\mathcal{F}, \mathcal{F}')$ soit suffisamment petite, nous pouvons éviter de construire tout $\mathcal{F}_x \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ tels que $d(\mathcal{F}_x, \mathcal{F}') \geq d(\mathcal{F}, \mathcal{F}')$. Notons que nous n'avons pas besoin d'obtenir tout $\mathcal{F}_x$ pour décider si $d(\mathcal{F}_x, \mathcal{F}') \geq d(\mathcal{F}, \mathcal{F}')$ car la construction de $\mathcal{F}_x$ est monotone. Si nous utilisons $\mathcal{F}(\mathcal{F}') \subseteq \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ pour désigner un ensemble de telles forêts de complétion $\mathcal{F}_x$ pour chaque $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$, il suffit de calculer une seule $\mathcal{F}_x$ dans $\mathcal{F}(\mathcal{F}')$. La figure 5.2 montre comment $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ est partitionné en des sous-ensembles $\mathcal{F}(\mathcal{F}'_i)$ pour chaque $\mathcal{F}'_i \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$.

En outre, le calcul de $\mathcal{F}(\mathcal{F}')$ est indépendant de $\mathcal{F}(\mathcal{F}'')$ avec $\mathcal{F}' \neq \mathcal{F}''$. Ceci nous permet de paralléliser le calcul de tout $\mathcal{F}(\mathcal{F}')$ avec $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$.

Une autre optimisation est liée à la taille des forêts de complétion. Il est avéré que la taille de chaque forêt de complétion est bornée par une fonction doublement exponentielle de la taille de l'ontologie. Une méthode présentée par Le Duc, Lamolle et Curé [Le Duc *et*

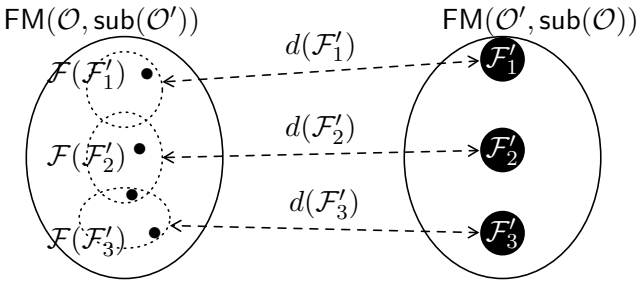


FIGURE 5.2 – Optimisation du calcul de $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$

al., 2013] permet de construire une structure, à savoir un *frame* compressant des nœuds similaires d'un arbre de complétion au lieu de construire tout l'arbre de complétion. Les auteurs [Le Duc *et al.*, 2013] ont démontré que la taille d'un *frame* est bornée par une fonction exponentielle (simple) de la taille de l'ontologie. Un avantage de cette méthode est que presque toutes les techniques d'optimisation conçues pour les algorithmes de tableau classiques restent utilisables.

5.5 Conclusion

Dans ce chapitre, nous avons introduit certaines techniques d'optimisation pour réduire la complexité des algorithmes lors de la révision d'ontologies. Ces techniques sont développées pour (i) réduire des non-déterminismes lors de l'application de l'algorithme de tableau, (ii) optimiser le temps du calcul de distance entre des modèles d'arbre ou entre des modèles de forêt, et (iii) éviter de construire des forêts ou des arbres non nécessaires à la révision. Nous avons appliqué ces techniques d'optimisation pour implémenter un prototype de révision d'ontologies que nous allons présenter dans le chapitre suivant.

Chapitre 6

Moteur de révision

Dans ce chapitre, nous présentons le prototype que nous avons implémenté pour la révision d'ontologies. Nous effectuerons quelques expérimentations avec de petites ontologies montrant le processus de révision d'ontologies. Il est à noter que ce prototype est utilisable en ligne afin que des utilisateurs puissent faire des tests via une interface web.

6.1 Prototype OntoRev

Nous avons mis en place un moteur de révision en tant que prototype, appelé OntoRev, qui est basé sur les algorithmes et les définitions décrits dans le chapitre précédent. De même que les raisonneurs LD tels que HermiT [Shearer *et al.*, 2008], Pellet [Sirin *et al.*, 2007], FaCT++ [Tsarkov and Horrocks, 2006], nous avons implémenté dans OntoRev diverses techniques d'optimisation telles que l'absorption, les blocages. Pour ce faire, nous avons utilisé des absorptions basiques et binaires (*cf. la section 5.1 du chapitre 5*) pour réduire les cas non déterministes résultant de la disjonction. Nous avons également appliqué la technique de blocage de noyau ("core blocking" en anglais) [Glimm *et al.*, 2010] à côté de la technique de blocage de paire ("pairwise blocking" en anglais) pour réduire la taille des forêts de complétion. Contrairement aux raisonneurs de tableau existants, nous devons explorer tous les cas intrinsèques non déterministes impliqués dans les ontologies pour construire toutes les forêts de complétion. En conséquence, nous considérons toujours

les scénarios du pire des cas où toutes les forêts seraient construites pour représenter la sémantique d'une ontologie. Dans la version actuelle d'OntoRev, certaines techniques d'optimisation telles que l'élagage des points de retour ("pruning of backtracking points") pour traiter le non-déterminisme intrinsèque n'ont pas été mises en œuvre. Notons que le manque d'implémentation d'optimisations avancées peut ralentir OntoRev lorsqu'il fonctionne sur des ontologies contenant une quantité importante de non-déterminisme.

La structure de donnée la plus importante de OntoRev est TREENODE dont les objets peuvent être combinés dans un arbre. Un tel arbre n'est pas un arbre spécial tel qu'un arbre équilibré. Il peut avoir n'importe quel nombre de niveaux et chaque nœud peut avoir n'importe quel nombre d'enfants. Chaque nœud de l'arbre peut avoir au plus un parent et de 0 à plusieurs enfants. TREENODE fournit des opérations pour examiner et modifier les parents et les enfants d'un nœud. Le sous-arbre enraciné en un nœud est l'ensemble de tous les nœuds qui peuvent être atteints en commençant par ce nœud et en suivant tous les liens possibles vers parents et enfants. Un nœud sans parent est la racine de son arbre ; un nœud sans enfant est une feuille. Un arbre peut être constitué de plusieurs sous-arbres ; et, chaque nœud agit comme la racine pour son propre sous-arbre.

A chaque objet de TREENODE est également assigné des attributs de validation, de saturation et d'application de sat-règle. L'algorithme de tableau applique des règles possibles sur chaque objet de TREENODE pour construire des arbres ou des forêts. De plus, en appliquant les techniques d'optimisation lors de la construction de $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$ (*cf. la section 5.3 du chapitre 5*), l'étiquette de chaque nœud (objet de TREENODE) d'une forêt dans $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$ doit être validée, saturée et la sat-règle appliquée si nécessaire.

Les étapes effectuées dans OntoRev sont illustrées dans la figure 6.1. Il y a quatre fonctions principales définies comme suit : CHO - chargeur d'ontologies ; CTM - constructeur de modèles de forêt ; CAR - calculateur de modèles résultants de révision ; et CTO - constructeur de l'ontologie de révision. Tout d'abord, la fonction CHO charge l'ontologie initiale et la nouvelle ontologie à réviser. Ensuite, les modèles de forêt et les modèles résultants de la révision (*i.e.*, $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$) sont construits par les fonctions CTM et CAR respectivement. Enfin, CTO construit l'ontologie de révision à partir de l'ensemble des forêts $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$. Nous allons présenter quelques expérimentations dans la section suivante.

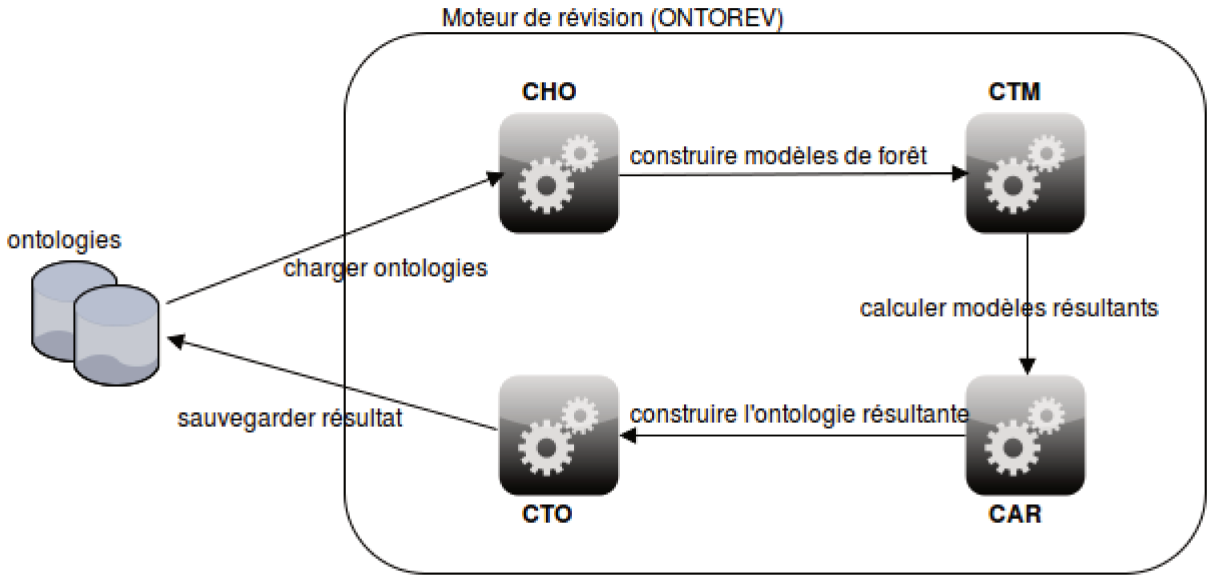


FIGURE 6.1 – Étapes effectuées dans ONTOREV

6.2 Expérimentation

Les expérimentations ont été effectuées sur les ontologies `GALEN`, `PIZZA` and `TRAINING`. La raison de ces choix est dû au fait que `PIZZA` est une petite ontologie avec une quantité importante de non-déterminisme résultant des disjonctions tandis que `GALEN` forcerait un algorithme de tableau à construire des forêts de complétion avec une profondeur importante. Enfin, `TRAINING` provient d'un projet de recherche FUI sur l'e-learning [1] qui inclut un moteur de révision au sein de sa plate-forme. Dans la suite de ce chapitre, nous présentons des expérimentations avec de nouvelles ontologies qui consistent en un axiome de concept, une assertion de concept et une assertion de rôle, respectivement.

• `GALEN_1` est obtenue de `GALEN` en ajoutant une instance d'un concept *PapillaryMuscle*. Ce concept est subsumé par *Muscle* et le concept complexe $\exists hasIntrinsicAbnormalityStatus.normal$ (chaque instance de *PapillaryMuscle* a un statut étant normal) et $\exists hasSurfaceVisibility.internal$ (chaque instance de *PapillaryMuscle* a une visibilité étant interne). Nous supposons que la connaissance de *PapillaryMuscle* change de telle sorte que nous devons ajouter les deux axiomes suivants :

$PapillaryMuscle \sqsubseteq \forall hasIntrinsicAbnormalityStatus.nonNormal$,

1. http ://www.omendo.com/plateforme-learning-cafe

Ontologie initiale	Caractéristiques de l'ontologie					
				Axiomes		
	Concepts	Rôles	Assertions	Inclusion	Équivalence	Disjoint
GALEN_1	2748	413	2	3238	699	2
GALEN_2	2748	413	1	3239	699	1
PIZZA	99	5	2	259	8	398
TRAINING	451	95	2	442	0	79

Nouvelle Ontologie	Caractéristiques de l'ontologie					
				Axiomes		
	Concepts	Rôles	Assertions	Inclusion	Équivalence	Disjoint
REV_GALEN_1	3	2	0	2	0	0
REV_GALEN_2	2	1	0	1	0	0
REV_PIZZA	4	1	0	2	0	0
REV_TRAINING	2	2	1	1	0	0

TABLEAU 6.1 – Ontologies pour les expérimentations et leurs caractéristiques

Ontologie	Résultat de Révision				
	$\lvert\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))\rvert$	$\lvert\mathsf{FM}(\mathcal{O}, \mathcal{O}')\rvert$	Profondeur d'arbre	N^0 de disjonctions	Temps (sec.)
REV_GALEN_1	1	1	3	11	3
REV_GALEN_2	1	1	6	17	4
REV_PIZZA	4096	4096	2	18	165
REV_TRAINING	2	2	1	4	2

TABLEAU 6.2 – Résultats d'expérimentations

$PapillaryMuscle \sqsubseteq \forall hasSurfaceVisibility.surfaceVisible$

qui constituent une nouvelle ontologie REV_GALEN_1. Notons que les concepts *nonNormal* et *surfaceVisible* sont disjoints avec *normal* et *internal* respectivement. Donc, GALEN_1 $\cup$ REV_GALEN_1 est incohérente. Le tableau 6.2 représente le résultat de l'opération de révision appliquée à l'ontologie initiale GALEN_1 avec la nouvelle ontologie REV_GALEN_1. $\mathsf{FM}(\texttt{GALEN\_1}, \mathsf{sub}(\texttt{REV\_GALEN\_1}))$ ne contient qu'une forêt de profondeur 3 puisque tout non-déterminisme non intrinsèque résultant de la $\sqsubseteq$-règle est absorbé et GALEN_1 ne

contient aucune disjonction.

Nous avons également effectué une autre expérimentation sur GALEN qui génère des forêts de complétion de plus grandes profondeurs. GALEN_2 est créée à partir de GALEN en ajoutant une instance de concept *Milk* qui est subsumé par *Substance* et le concept complexe $\exists hasState.liquid$ (l'état d'une instance de *Milk* peut être liquide). Supposons que la connaissance de *Milk* évolue de telle sorte que l'état de chaque instance de *Milk* soit solide. Cette nouvelle connaissance peut être formulée par $Milk \sqsubseteq \forall hasState.solid$; ce qui constitue une nouvelle ontologie REV_GALEN_2. Notons que *solid* est disjoint avec *liquid*. Donc, il est clair que GALEN_2 $\cup$ REV_GALEN_2 est incohérente. Le tableau 6.2 représente le résultat de l'opération de révision appliquée à l'ontologie initiale GALEN_2 avec la nouvelle ontologie REV_GALEN_2. Nous observons que $\mathsf{FM}(\texttt{GALEN\_2}, \mathsf{sub}(\texttt{REV\_GALEN\_2}))$ ne contient qu'une forêt de profondeur 6.

• Comme mentionné ci-dessus, la version actuelle d'OntoRev ne se comporte pas de façon optimale sur une ontologie contenant une quantité importante de non-déterminisme. Nous avons déterminé ce problème en expérimentant notre moteur avec une ontologie PIZZA du monde réel qui a beaucoup de cas non-déterministes. Pour ce faire, nous avons simplifié l'ontologie initiale PIZZA en enlevant certains axiomes d'équivalence/individus qui sont responsables des non-déterminismes "difficiles". A l'heure actuelle, ce retrait nous permet de réduire le temps de construction ainsi que la taille de l'ensemble des forêts de complétion. Dans l'ontologie modifiée PIZZA, le concept *Fiorentina* est subsumé par *Pizza* et le concept complexe $\exists hasTopping.SpinachTopping$ (*i.e.* les ingrédients de garniture d'une instance de *Fiorentina* peuvent être des épinards). De plus, le concept *Soho* est subsumé par le concept *Pizza* et le concept complexe $\exists hasTopping.GarlicTopping$ (*i.e.* les ingrédients de garniture d'une instance de *Soho* peuvent être des aulx). Nous supposons que la connaissance de *Fiorentina* et *Soho* change de telle sorte que les ingrédients de garniture de *Fiorentina* ne doivent pas être des épinards et ceux de *Soho* ne doivent pas être des aulx. Cette nouvelle connaissance peut être formulée comme $Fiorentina \sqsubseteq \forall hasTopping.(\neg SpinachTopping)$ et $Soho \sqsubseteq \forall hasTopping.(\neg GarlicTopping)$ qui constituent une nouvelle ontologie REV_PIZZA. Donc, PIZZA $\cup$ REV_PIZZA est incohérente. Le tableau 6.2 représente le résultat de l'opération

de révision appliquée à l'ontologie modifiée PIZZA avec la nouvelle ontologie REV_PIZZA. Même en ayant enlevé certains non-déterminismes dans l'ontologie initiale PIZZA, nous observons que $\mathsf{FM}(\texttt{PIZZA}, \mathsf{sub}(\texttt{REV\_PIZZA}))$ contient toujours plusieurs forêts de complétion. Cet inconvénient peut être expliqué par l'existence d'autres disjonctions contenues dans les axiomes nous forçant à créer plusieurs copies de forêts.

• L'ontologie TRAINING représente le domaine de l'apprentissage de gestes techniques pour des métiers manuels. L'acquisition de ces compétences est basées sur des ressources pédagogiques multimédia. Dans cette ontologie, il y a une instance *bkTr* qui satisfait le concept *BakerTraining*. Ce concept définit que toute formation en boulangerie a un objet d'apprentissage ($BakerTraining \sqsubseteq \exists hasLearningObject.LearningObject$). Cela implique que le co-domaine de *hasLearningObject* contient au moins un individu de *LearningObject*. De plus, TRAINING impose également que le co-domaine du rôle *hasLearningObject* est le concept *PedagogicalResource*, et *PedagogicalResource* est disjoint avec le concept *LearningObject*. Nous souhaitons maintenant que l'instance *bkTr* a un objet d'apprentissage *PR1* qui est une instance du concept *PedagogicalResource*, *i.e.*, *PedagogicalResource(PR1)* et $hasLearningObject(bkTr, PR1)$. Il en résulte une incohérence. Donc, REV_TRAINING se compose d'un axiome et une assertion formulés à partir des nouvelles connaissances ci-dessus ($BakerTraining \sqsubseteq \exists hasLearningObject.PedagogicalResource$, et $hasLearningObject(bkTr, PR1)$).

Nous présentons dans le tableau 6.1 les caractéristiques des ontologies utilisées pour les tests et dans le tableau 6.2 les résultats obtenus. Nous avons effectué tous les tests sur un DELL avec 8 Processeurs Intel de 3.4GHz et 32Gb de RAM sous Ubuntu. Comme mentionné dans la section 5.3, la construction de $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ sans la sat-règle nous permet d'explorer uniquement les cas non déterministes intrinsèques dans les ontologies. L'ensemble $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ peut aider à construire $\mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ puisqu'il contient des candidats de forêts qui ont des distances minimales à $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$. Alors, la taille de $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$ est supérieur ou égal à celle de $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$. De plus, il peut exister des forêts de complétion dans $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$ qui sont équivalentes. Cela explique pourquoi le nombre de disjonctions (N^0 de disjonctions dans le tableau 6.2) dans l'axiome de l'ontologie résultante $\mathcal{O}^*$ est petit. En outre, nous pouvons également restaurer des axiomes à partir de

l'ontologie initiale en vérifiant si un nom de concept se trouve dans une étiquette de nœud d'une forêt de complétion ; si ce n'est pas le cas, nous ajoutons directement à l'ontologie résultante les axiomes transformés par absorption.

6.3 Utilisation et mise en ligne d'OntoRev

Comme mentionné dans l'introduction générale, cette thèse s'intègre dans le projet *LearningCafé* (FUI). Un premier objectif de la révision d'ontologies dans le cadre de ce projet a été de mettre à jour des connaissances modélisées dans les ontologies. La nécessité d'une mise à jour peut provenir par exemple du fait que : (i) une vidéo vient d'être ajoutée, (ii) une formation vient d'être modifiée ou ajoutée, (iii) les informations sur le profil d'un utilisateur sont mises à jour. De tels changements devraient être validés par un groupe d'utilisateurs restreints (*e.g.* l'administrateur des ontologies, les formateurs, etc.) car ils peuvent se répercuter sur d'autres modules de la plate-forme qui utilisent les ontologies.

Un module (*cf. le rectangle "Moteur de révision" dans la figure 6.2*) chargé de la révision d'ontologie (OntoRev) est intégré dans la plate-forme *LearningCafé*. Ce module reçoit une requête de révision de l'interface de la plate-forme (*cf. la flèche 1.2 dans la figure 6.2*) et effectue la révision sur les ontologies concernées de la plate-forme. Une telle révision peut entraîner une modification des ontologies en assurant leur cohérence. Un message est renvoyé à la plate-forme pour informer du résultat de la requête de révision. Tous ces échanges s'effectuent par des services Web et sont illustrés dans la figure 6.2.

Nous avons implémenté et mis en ligne OntoRev [2] un module simple pour la mise à jour des connaissances modélisées dans les ontologies.

La page d'accueil (*cf. figure 6.3*) est découpée en trois zones :

— **Ontology Information** : Un utilisateur peut charger une ontologie en tapant un lien [3] de l'ontologie ou en choisissant (**Browse...**) une ontologie en local. Notons que si le bouton **Load** situé dans la zone **Toolbox** est cliqué mais que l'utilisateur

2. http ://linc.iut.univ-paris8.fr :8080/search-revision-engine/
3. sous la forme de http ://exemple.com/url/vers/votreOntologie.owl

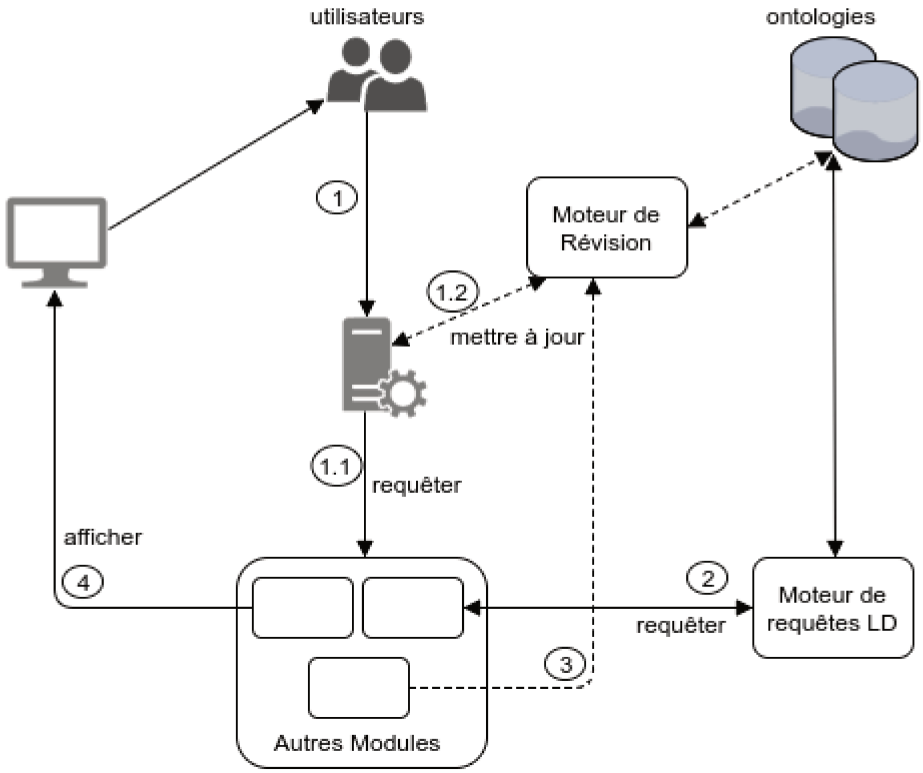


FIGURE 6.2 – Échanges de données entre le module de révision et d'autres modules de la plate-forme *LearningCafé*

n'a pas tapé de lien ni choisi une ontologie en local, l'ontologie *Training.owl* de la plate-forme *LearningCafé* se charge par défaut.

— **Toolbox** : contient des boutons qui permettent de charger une ontologie (**Load**), de sauvegarder le résultat (**Save**), ou d'annuler le traitement (**Cancel**). De plus, l'utilisateur peut cliquer sur **More Infos** pour obtenir plus d'information sur l'ontologie en cours de traitement. Le bouton **Add New Infos** lui permet de mettre à jour l'ontologie. Enfin, il peut aussi faire des requêtes sur l'ontologie en utilisant le bouton **DL Query & Entailment**.

— **Console** : affiche une notification lors d'une action qui est réalisée sur le moteur.

En outre, une fois qu'une ontologie est chargée, notre moteur de révision affichera de nouveaux boutons (*cf. figure 6.4*) qui permettent d'ajouter de nouvelles entités, de nouveaux axiomes ou de nouvelles assertions.

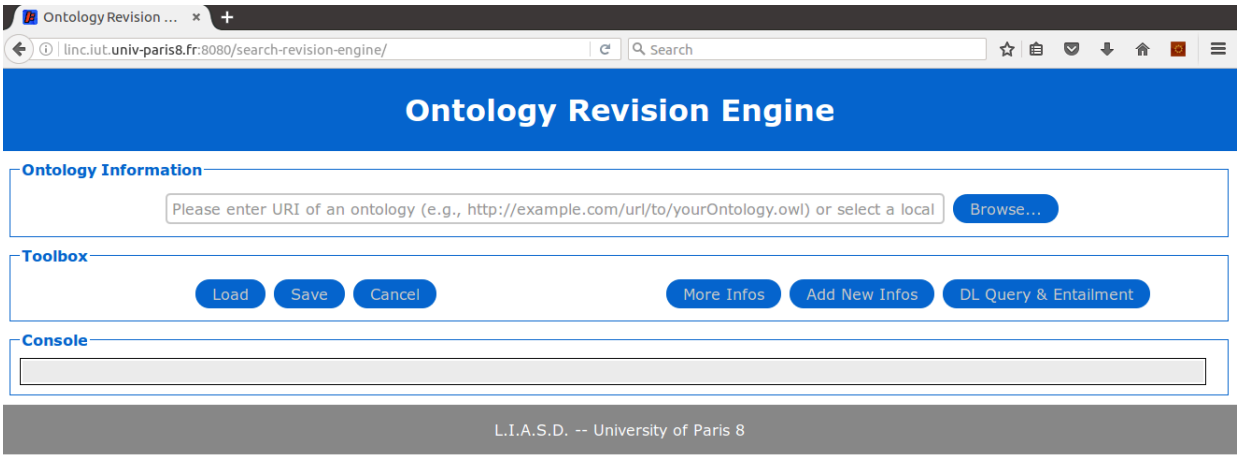


FIGURE 6.3 – Page d'accueil

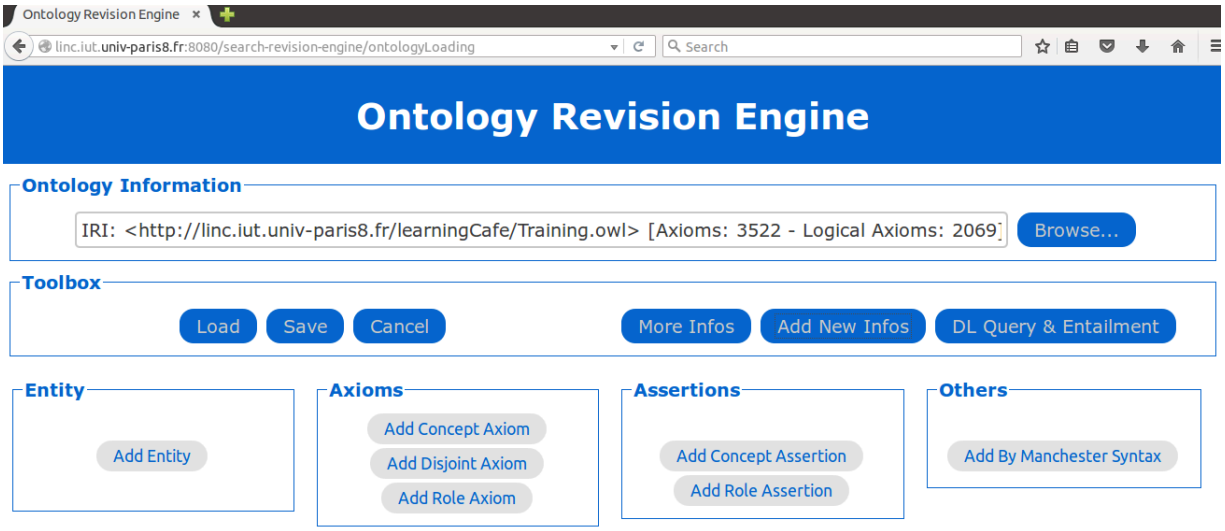


FIGURE 6.4 – Outils pour la révision d'une ontologie

6.4 Conclusion

Nous avons implémenté et testé un prototype OntoRev pour la révision d'ontologies en $\mathcal{SHIQ}$. Les résultats expérimentaux obtenus ont indiqué que notre algorithme / implémentation n'est pas encore assez performant sur des ontologies qui contiennent une quantité importante de non-déterminisme résultant de disjonctions, de restrictions de cardinalité. Pour résoudre ces inconvénients, nous chercherons à concevoir un algorithme de tableau ExpTime pour vérifier la cohérence des ontologies en $\mathcal{SHIQ}$ car un tel algorithme nous permettrait de réduire non seulement le non-déterminisme, mais aussi la profondeur des

forêts de complétion. Nous avons aussi implémenté et exécuté en ligne [4] un moteur (intégrant le prototype OntoRev) pour la mise à jour des connaissances modélisées dans les ontologies.

4. http ://linc.iut.univ-paris8.fr :8080/search-revision-engine/

Conclusion

Nous avons eu l'occasion lors de ce mémoire d'exposer la problématique de la révision d'ontologie et plus particulièrement celle inhérente aux ontologies exprimée en logique de description $\mathcal{SHIQ}$. Rappelons que la révision consiste à ajouter une nouvelle connaissance à une ontologie et faire les changements nécessaire pour que l'ontologie reste cohérente. Pour cela, nous avons tout d'abord fait une étude approfondie des approches existantes à savoir les approches syntaxiques et sémantiques dans le domaine de la révision d'ontologies. Les approches syntaxiques souffrent de deux limitations majeures à savoir, d'une part, la non unicité de résultat, *i.e.* l'existence de plusieurs ontologies résultantes avec des conséquences logiques différentes, d'autre part, le changement excessif, *i.e.* le risque de supprimer des connaissances qui sont compatibles avec la nouvelle connaissance.

Cette étude a également présenté plusieurs méthodes de révision sémantique qui utilisent différentes structures pour représenter la sémantique d'une ontologie. Cependant, ces méthodes visent à traiter les ontologies inexpressives et ne garantissent pas le changement minimal effectué sur une ontologie pour qu'elle reste cohérente après révision.

Pour cette raison, nous avons adopté une approche sémantique fondée sur une caractérisation de la sémantique d'ontologie en utilisant un ensemble de modèles les plus représentatifs de l'ontologie. Notre approche de révision consiste donc à développer un nouvel algorithme de tableau permettant de construire les modèles sous forme de graphes finis (appelés *modèles d'arbre* pour les ontologies sans individu ou *modèles de forêt* pour les ontologies avec individus) pour représenter un ensemble éventuellement infini de modèles d'une ontologie. La particularité de cet algorithme de tableau est qu'il doit construire tous les modèles de forêt qui correspondent aux non-déterminismes intrinsèques présents dans l'ontologie.

Dans ce cas, le problème de révision d'une ontologie est réductible à celui du calcul d'un ensemble de modèles de forêt représentant la sémantique de l'ontologie résultante. La procédure pour ce calcul nécessite la notion de distance entre modèles de forêt qui implique un pré-ordre total sur les modèles de forêt. Ce pré-ordre nous permet de choisir, parmi les modèles de forêts possibles vérifiant la nouvelle connaissance, un ensemble de modèles les plus appropriés pour assurer le changement minimal, et donc les postulats AGM, *i.e.* les modèles choisis devraient être les plus proches sémantiquement de ceux de l'ontologie initiale. Grâce à cette distance, nous avons défini une opération de révision d'ontologies qui détermine précisément l'ensemble des modèles que l'ontologie résultante devrait admettre. Cependant, il n'est pas nécessaire d'exister une ontologie en $\mathcal{SHIQ}$ qui admette exactement comme modèles un ensemble de graphes calculé par l'opération de révision. Afin de remédier à ce problème, nous avons proposé une construction de l'ontologie sémantiquement la plus petite en $\mathcal{SHIQ}$ qui admette un ensemble de graphes comme modèles. Enfin, nous avons aussi implémenté notre algorithme de révision dans un prototype baptisé OntoRev avec les techniques d'optimisation que nous avons élaborées telles que l'absorption, le calcul polynomial de la distance entre des graphes. Des expérimentations ont également été effectuées pour montrer que notre algorithme est utilisable en pratique.

Même si notre prototype révèle certaines limites (*e.g.*, la complexité de la construction des modèles de graphe, la taille triplement exponentielle de l'ontologie de révision), notre travail fournit, à notre connaissance, un des premiers résultats pour la révision des ontologies en $\mathcal{SHIQ}$.

Les perspectives de ces travaux à plus ou moins long terme vont aborder les points suivants :

1. Amélioration de l'implémentation en appliquant des techniques d'optimisation existantes dans l'algorithme de tableau standard. Ces techniques d'optimisation sont implémentées en pratique dans les raisonneurs LD tels que HermiT [Shearer *et al.*, 2008], Pellet [Sirin *et al.*, 2007], FaCT++ [Tsarkov and Horrocks, 2006].

   Technique de branchement sémantique. En présence d'une disjonction, il existe peut-être plusieurs branches ouvertes à la recherche. La méthode naïve pour satisfaire une disjonction $C \sqcup D$ est d'ajouter C d'abord et, si cela cause un clash, ajouter D

par la suite. Ceci est plutôt inefficace parce que les ressources ont été dépensées pour trouver que C n'est pas satisfiable sur un nœud dans l'arbre actuel mais cette information est oubliée. Le branchement sémantique [Freeman, 1995; Horrocks, 2003] ajoute $\neg C \sqcap D$ dans l'étiquette du nœud si C cause un clash. Cela rend explicite le fait que l'information C n'est pas satisfiable et que l'on peut éventuellement élaguer l'espace de recherche car le nœud de l'arbre dans lequel C est insatisfiable n'est jamais testé par la suite.

Backtracking. Lors d'un clash, le *backtracking* naïf revient au point de branchement le plus récent. Le *backtracking en fonction de la dépendance* (*"backjumping"*) [Horrocks, 2003] est une technique de backtracking optimisée qui permet à l'algorithme de tableau de revenir en arrière vers la partie concernée plutôt que le point de branchement le plus récent. Dans ce cas, les points de branchement intermédiaires, qui n'ont eu aucune influence sur le clash, sont ignorés.

Propagation de contrainte booléenne [Freeman, 1995]. Avant de choisir un élément C d'une disjonction $C \sqcup D$ dans l'étiquette d'un nœud x pour créer un nouveau branchement pour C, chaque élément est vérifié pour savoir si sa négation a déjà été inclue dans l'étiquette de x. Dans ce cas, cet élément est *fermé*, sinon il est *ouvert*. Si C est fermé, il est enlevé de la disjonction et seuls les éléments restants de la disjonction sont pris en compte pour le branchement. En particulier, le branchement n'est pas nécessaire s'il n'y a qu'un seul élément ouvert car cet élément peut être ajouté tout de suite dans l'étiquette de x sans clash.

2. Extension de l'algorithme pour la révision des ontologies exprimées dans des logiques plus expressives, *e.g.* $\mathcal{SHOIQ}$. La logique $\mathcal{SHOIQ}$ est une extension de $\mathcal{SHIQ}$ en ajoutant des nominaux $\{a\}$. Cette extension peut poser certaines difficultés à l'algorithme de tableau s'appliquant sur des ontologies en $\mathcal{SHOIQ}$. Les difficultés sont dues à l'interaction entre les nominaux, les restrictions de cardinalité et les rôles inverse, ce qui entraîne la perte presque complète de la propriété de l'arbre représentant le modèle et provoque la complexité du problème de la cohérence de l'ontologie en passant de EXPTIME à NEXPTIME [Tobies, 2000].

3. Application de notre algorithme à la révision d'un réseau d'ontologies alignées. Eu-

zenat [Euzenat, 2015] a introduit des postulats pour faire la révision sur un réseau d'ontologies et a montré qu'une révision globale ne peut être réduite aux révisions locales en général. Ce résultat peut être considéré comme une conséquence du fait que les incohérences de l'ontologie et/ou les insatisfiabilités d'un concept peuvent être propagées d'une ontologie locale à une autre par des alignements. Nous pensons qu'il pourrait exister des restrictions sur l'expressivité de l'alignement qui permettent de réduire une révision sur un réseau d'ontologies aux révisions locales. Dans ce cas, la révision sur un réseau d'ontologies consisterait à réviser des ontologies locales et à propager des connaissances entre les ontologies locales avant et après la révision des ontologies locales.

Annexe

Lemme 2. ***(Terminaison, Correction et Complétude).***

Soit $\mathcal{O}$ une ontologie en $\mathcal{SHIQ}$.

1. *L'algorithme 1 se termine.*
2. *Si l'algorithme 1 peut être appliqué à $\mathcal{O}$ de telle façon qu'il construise un arbre de complétion non-clash et complet alors $\mathcal{O}$ est cohérente ;*
3. *si $\mathcal{O}$ est cohérente alors l'algorithme 1 peut être appliqué à $\mathcal{O}$ de telle façon qu'il construise un arbre de complétion non-clash et complet.*

Démonstration.

1. Soit $m = |\mathsf{sub}(\mathcal{O})|$, $k = |\mathbf{R} \cup \mathbf{R}_\mathsf{I}|$. La terminaison de l'Algorithme 1 est montrée par les propriétés suivantes des règles d'expansion :

   (a) L'application des règles de la Figure 3.1 ne supprime jamais de concepts depuis des étiquettes de nœud. Les étiquettes d'arête ne peuvent être changées par la $\leq$-règle qui soit les étend soit les rend vide (*i.e.* $\emptyset$), et lorsque l'étiquette d'une arête devient vide, elle reste toujours vide.

   (b) Les successeurs d'un nœud x doivent être le résultat d'une application des $\exists$- ou $\geq$-règle aux concepts sous la forme de $\exists S.C$ (qui génère un successeur) et de $(\geq nS.C)$ (qui génère n successeurs) dans $\mathcal{L}(x)$. Pour un nœud x, chacun de ces concepts peut déclencher la génération de successeurs au plus une fois. Pour la $\exists$-règle, si un successeur y de x est généré pour un concept $\exists S.C \in \mathcal{L}(x)$ et $\mathcal{L}(\langle x, y\rangle)$ est fixé à $\emptyset$ par une application de la $\leq$-règle, alors il y a des S-voisins

z de x tel que $X \subseteq \mathcal{L}(z)$ pour des $X \in \mathsf{Flat}(C)$. Pour la $\geq$-règle, si $y_1, \ldots, y_n$ sont générés par une application de la $\geq$-règle pour un concept $(\geq nS.C)$, alors $y_i \not\equiv y_j$ est vrai pour tout $1 \leq i < j \leq n$. Cela implique qu'il y ait n S-voisins $y'_1, \ldots, y'_n$ de x avec $X \subseteq \mathcal{L}(y'_i)$ et $y'_i \not\equiv y'_j$ pour certains $X \in \mathsf{Flat}(C)$ et tout $1 \leq i < j \leq n$, car la $\leq$-règle ne peut jamais fusionner deux nœuds y'_i, y'_j (car $y'_i \not\equiv y'_j$), et, une fois que l'application de la $\leq$-règle fixe $\mathcal{L}(\langle x, y'_i \rangle)$ à $\emptyset$, alors il y a des S-voisins z de x avec $X \subseteq \mathcal{L}(z)$ pour certains $X \in \mathsf{Flat}(C)$ et z hérite de toutes les inégalités de y'_i.

Comme $\mathsf{sub}(\mathcal{O})$ contient au plus m concepts $\exists S.C$ et $(\geq nS.C)$, le degré sortant de l'arbre est borné par $m \times n_{max}$ où n_{max} est le n maximum qui se trouve dans un concept sous la forme de $(\bowtie nS.C) \in \mathsf{sub}(\mathcal{O})$.

(c) Les nœuds sont étiquetés par des sous-ensembles non-vides de $\mathsf{sub}(\mathcal{O})$ et les arêtes sont étiquetées par des sous-ensembles non-vides de $\mathbf{R} \cup \mathbf{R_I}$; il y a donc au plus $K = 2^{2mk}$ étiquettes différentes possibles pour une paire de nœuds et une arête. Par conséquent, selon la condition de blocage, si un chemin est d'une longueur d'au moins K alors il existe deux nœuds x, y tels que x est bloqué par y. Comme un chemin sur lequel les nœuds sont bloqués ne peut pas devenir plus long, les chemins sont d'une longueur d'au plus K.

2. Nous démontrons que l'application de l'Algorithme 1 à une ontologie $\mathcal{O}$ construisant un arbre de complétion non-clash et complet implique la cohérence de $\mathcal{O}$. Soit T un arbre de complétion non-clash et complet, un *chemin* est une séquence de paire de nœuds de T sous la forme $p = [\frac{x_0}{x'_0}, \ldots, \frac{x_n}{x'_n}]$. Pour chaque chemin, nous définissons $\mathsf{Tip}(p) := x_n$ et $\mathsf{Tip'}(p) := x'_n$. Avec $[p|\frac{x_{n+1}}{x'_{n+1}}]$, nous désignons le chemin $[\frac{x_0}{x'_0}, \ldots, \frac{x_n}{x'_n}, \frac{x_{n+1}}{x'_{n+1}}]$. L'ensemble $\mathsf{Paths}(T)$ est défini de manière inductive comme suit :

• Pour le nœud racine $\widehat{x} = x_0$ de T, $[\frac{x_0}{x_0}] \in \mathsf{Paths}(T)$, et

• Pour chaque chemin $p \in \mathsf{Paths}(T)$ et chaque successeur z de $\mathsf{Path}(p)$ dans T :

— si z n'est pas bloqué alors $[p|\frac{z}{z}] \in \mathsf{Paths}(T)$, sinon

— si z est bloqué par un nœud y alors $[p|\frac{y}{z}] \in \mathsf{Paths}(T)$.

Notons que, par construction de $\mathsf{Paths}(T)$ et la condition de blocage,

(a) si $p \in \mathsf{Paths}(T)$ alors $\mathsf{Tip}(p)$ n'est pas bloqué,

(b) $\mathsf{Tip}(p) = \mathsf{Tip}'(p)$ si et seulement si $\mathsf{Tip}'(p)$ n'est pas bloqué, et

(c) $L(\mathsf{Tip}(p)) = L(\mathsf{Tip}'(p))$.

Nous définissons une interprétation $\mathcal{I} = (\Delta, .^{\mathcal{I}})$ comme suit :

- $\Delta := \mathsf{Paths}(\mathcal{F})$,
- $A^{\mathcal{I}} := \{p \mid A \in L(\mathsf{Tip}(p))\}$ pour tout nom de concept A dans $\mathsf{sub}(O)$,
- $R^{\mathcal{I}} := \begin{cases} \mathcal{E}(R)^+ \text{ si } \mathsf{Trans}(R) = \mathsf{vrai} \\ \mathcal{E}(R) \cup \bigcup\limits_{P \sqsubseteq R, P \neq R} P^{\mathcal{I}} \text{ sinon} \end{cases}$

où $\mathcal{E}(R)^+$ est la fermeture transitive de $\mathcal{E}(R) := \{\langle p, [p|\frac{x}{x'}]\rangle \in \Delta \times \Delta \mid x' \text{ est un } R\text{-successeur de } \mathsf{Tip}(p)\} \cup \{\langle [q|\frac{x}{x'}], q\rangle \in \Delta \times \Delta \mid x' \text{ est un } \mathsf{Inv}(R)\text{-successeur de } \mathsf{Tip}(q)\} \cup \{\langle [\frac{x}{x}], [\frac{y}{y}]\rangle \in \Delta \times \Delta \mid x, y \text{ sont des nœuds racines, et } y \text{ est un } R\text{-voisin de } x\}$. L'interprétation récursive des rôles non transitifs est nécessaire pour interpréter correctement les rôles non transitifs ayant un sous-rôle transitif.

Afin de montrer que $\mathcal{I} = (\Delta, .^{\mathcal{I}})$ est un modèle de $\mathcal{O}$, nous devons montrer que (i) $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ pour chaque $R \sqsubseteq S \in \mathcal{R}$, (ii) $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ pour chaque $C \sqsubseteq D \in \mathcal{T}$, (iii) $a^{\mathcal{I}} \in C^{\mathcal{I}}$ et $\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$ pour chaque $C(a) \in \mathcal{A}$ et $R(a, b) \in \mathcal{A}$ respectivement.

(i) $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ pour chaque $R \sqsubseteq S \in \mathcal{R}$. Supposons que $\mathsf{Trans}(R) = \mathsf{false}$. Selon l'application des règles sur $\mathcal{F}$, si $R \in L(\langle x, y\rangle)$ alors $S \in L(\langle x, y\rangle)$ où x, y sont deux nœuds de $\mathcal{F}$. En raison de la définition de $R^{\mathcal{I}}$ et $S^{\mathcal{I}}$, nous avons $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$. Supposons que $\mathsf{Trans}(R) = \mathsf{true}$. Si $\mathsf{Trans}(S) = \mathsf{false}$, nous avons $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ par la définition de $S^{\mathcal{I}}$; autrement, $\mathsf{Trans}(S) = \mathsf{true}$. Puisque l'application des règles sur $\mathcal{F}$ impose que si $R \in L(\langle x, y\rangle)$ alors $S \in L(\langle x, y\rangle)$ où x, y sont deux nœuds de $\mathcal{F}$, nous avons aussi $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ par la définition de $R^{\mathcal{I}}$ et $S^{\mathcal{I}}$.

(ii) $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ pour chaque $C \sqsubseteq D \in \mathcal{T}$. Selon la sat-règle s'appliquant sur chaque nœud x de $\mathcal{F}$, il existe toujours $X \in \mathsf{Flat}(\neg C \sqcup D)$ tel que $X \subseteq L(x)$. En raison de la définition de Flat, nous pouvons écrire $X = Y \cup Z$ où $Y \in \mathsf{Flat}(\neg C)$ et $Z \in \mathsf{Flat}(D)$. Soit $\bar{X} = X_1 \sqcap \cdots \sqcap X_n$ où $X_i \in X$ $(1 \leq i \leq n)$, $\bar{Y} = Y_1 \sqcap \cdots \sqcap Y_m$ où $Y_j \in Y$ $(1 \leq j \leq m)$ et $\bar{Z} = Z_1 \sqcap \cdots \sqcap Z_k$ où $Z_t \in Z$ $(1 \leq t \leq k)$. Cela implique qu'il existe $p \in \bar{X}^{\mathcal{I}} = \bar{Y}^{\mathcal{I}} \cup \bar{Z}^{\mathcal{I}} \subseteq (\neg C)^{\mathcal{I}} \cup D^{\mathcal{I}}$ où $\mathsf{Tip}(p) = x$. Par conséquent, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

3. Il existe un modèle $\mathcal{I} = \langle \Delta, .^{\mathcal{I}} \rangle$ tel que $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, $a^{\mathcal{I}} \in C^{\mathcal{I}}$ et $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$ pour tous les axiomes $C \sqsubseteq D$. Cela veut dire que tous les sous-concepts X de l'ontologie $\mathcal{O}$ doivent être interprétés par $\mathcal{I}$ (sinon nous ne pourrions pas savoir si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ ou $a^{\mathcal{I}} \in C^{\mathcal{I}}$ avec X se produit dans C ou D). Donc, X est interprété par $\mathcal{I}$ pour chaque $X \in \mathsf{sub}(\mathcal{O})$. Alors, le modèle $\mathcal{I}$ de $\mathcal{O}$ est un modèle d'une ontologie $\mathcal{O}' = \mathcal{O} \cup \{X \sqsubseteq X \mid X \in \mathsf{sub}(\mathcal{O})\}$.

   Nous utilisons ce modèle pour déclencher l'application des règles. Pour ce faire, nous définirons inductivement une fonction π associant les nœuds de T à Δ comme suit :

   — Si $\pi(x) = s$ est déjà défini, et un successeur y de x est généré pour $\exists R.C \in L(x)$, alors $\pi(y) = t$ pour certains $t \in \Delta$ avec $X \subseteq L(t)$ et $\langle s, t \rangle \in R^{\mathcal{I}}$ pour certains $X \in \mathsf{Flat}(C)$.

   — Si $\pi(x) = s$ est déjà défini, et un successeur y de x est généré pour $(\geq nR.C) \in L(x)$, alors $\pi(y_i) = t_i$ pour n distincts $t_i \in \Delta$ avec $X \subseteq L(t_i)$ et $\langle s, t_i \rangle \in R^{\mathcal{I}}$ pour certains $X \in \mathsf{Flat}(C)$,

   et l'association de l'arbre de complétion T satisfaisant les conditions suivantes pour chaque x, y dans T :

$$\left.\begin{array}{l} L(x) \subseteq L(\pi(x)) = \{C \mid (\pi(x))^{\mathcal{I}} \in C^{\mathcal{I}}\} \\ \text{si } y \text{ est un S-voisin de } x \text{ alors} \langle (\pi(x))^{\mathcal{I}}, (\pi(y))^{\mathcal{I}} \rangle \in S^{\mathcal{I}} \\ x \neq y \Rightarrow (\pi(x))^{\mathcal{I}} \neq (\pi(y))^{\mathcal{I}} \end{array}\right\} (*)$$

   Affirmation : Soient T un arbre de complétion et π une fonction qui satisfait (*). Si une règle est applicable à T alors la règle est applicable à T afin d'avoir un arbre de complétion T' et une fonction π' qui satisfait (*).

   Soient T un arbre de complétion et π une fonction qui satisfait (*), nous devons considérer les règles suivantes :

   • La $\exists$-règle : si $\exists S.C \in L(x)$ alors $\exists S.C \in L(\pi(x))$ et $(\pi(x))^{\mathcal{I}} \in (\exists S.C)^{\mathcal{I}}$. Selon la définition de $(\exists S.C)^{\mathcal{I}}$, il y a un individu $t \in \Delta$ tel que $\langle (\pi(x))^{\mathcal{I}}, t^{\mathcal{I}} \rangle \in S^{\mathcal{I}}$ et $t^{\mathcal{I}} \in C^{\mathcal{I}}$. L'application de la $\exists$-règle génère une nouvelle variable y avec $L(\langle x, y \rangle) = \{S\}$ et $L(y) = X$ pour certains $X \in \mathsf{Flat}(C)$ tel que $t^{\mathcal{I}} \in X^{\mathcal{I}} \subseteq C^{\mathcal{I}}$. Donc, nous fixons $\pi' := \pi[y \to t]$ donnant une fonction qui satisfait (*) pour l'arbre modifié.

• La $\forall$-règle : si $\forall S.C \in L(x)$ alors $\forall S.C \in L(\pi(x))$, et $(\pi(x))^{\mathcal{I}} \in (\forall S.C)^{\mathcal{I}}$. De plus, si y est un S-voisin de x alors $\langle(\pi(x))^{\mathcal{I}}, (\pi(y))^{\mathcal{I}}\rangle \in S^{\mathcal{I}}$ selon (*). En raison de la définition de $(\forall S.C)^{\mathcal{I}}$, nous avons $(\pi(y))^{\mathcal{I}} \in X^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ pour des $X \in \mathsf{Flat}(C)$. Cela implique que la $\forall$-règle peut être appliquée sans violer (*).

• La $\forall_+$-règle : si $\forall S.C \in L(x)$ alors $\forall S.C \in L(\pi(x))$ et $(\pi(x))^{\mathcal{I}} \in (\forall S.C)^{\mathcal{I}}$. De plus, s'il existe un certain R avec $R \sqsubseteq S$ et $\mathsf{Trans}(R)$ et y est un R-voisin de x alors $\langle(\pi(x))^{\mathcal{I}}, (\pi(y))^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$ selon (*). Supposons que $(\pi(y))^{\mathcal{I}} \notin (\forall R.C)^{\mathcal{I}}$. Selon la définition de $(\forall R.C)^{\mathcal{I}}$, il existe $\langle(\pi(y))^{\mathcal{I}}, t^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$ telle que $t^{\mathcal{I}} \notin C^{\mathcal{I}}$. Puisque $R \sqsubseteq S$ et $\mathsf{Trans}(R)$, nous avons $\langle(\pi(x))^{\mathcal{I}}, (\pi(y))^{\mathcal{I}}\rangle, \langle(\pi(y))^{\mathcal{I}}, t^{\mathcal{I}}\rangle, \langle(\pi(x))^{\mathcal{I}}, t^{\mathcal{I}}\rangle \in R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$. Cela implique que $\langle(\pi(x))^{\mathcal{I}}, t^{\mathcal{I}}\rangle \in S^{\mathcal{I}}$ et $t^{\mathcal{I}} \notin C^{\mathcal{I}}$ contredit $(\pi(x))^{\mathcal{I}} \in (\forall S.C)^{\mathcal{I}}$. Alors $(\pi(y))^{\mathcal{I}} \in (\forall R.C)^{\mathcal{I}}$ et donc la $\forall_+$-règle peut être appliquée sans violer (*).

• La $\geq$-règle : si $(\geq nSC) \in L(x)$ alors $(\geq nSC) \in \mathcal{L}(\pi(x))$ et $(\pi(x))^{\mathcal{I}} \in (\geq nS.C)^{\mathcal{I}}$. Selon la définition de $(\geq nS.C)^{\mathcal{I}}$, il existe des individus $t_1, \ldots, t_n \in \Delta$ tels que $\langle(\pi(x))^{\mathcal{I}}, t_i^{\mathcal{I}}\rangle \in S^{\mathcal{I}}$, $t_i^{\mathcal{I}} \in C^{\mathcal{I}}$ et $t_i^{\mathcal{I}} \neq t_j^{\mathcal{I}}$ pour $1 \leq i < j \leq n$. La $\geq$-règle génère n nouveaux nœuds $y_1, \ldots, y_n$ avec $L(\langle x, y_i\rangle) = \{S\}$ et $L(y_i) = X$ pour certains $X \in \mathsf{Flat}(C)$ tel que $t_i^{\mathcal{I}} \in X^{\mathcal{I}} \subseteq C^{\mathcal{I}}$. En fixant $\pi' := \pi[y_1 \to t_1, \ldots, y_n \to t_n]$, nous obtenons une fonction π' qui satisfait (*) pour l'arbre modifié.

• La $\leq$-règle : si $(\leq nSC) \in L(x)$ alors $(\leq nSC) \in L(\pi(x))$ et $(\pi(x))^{\mathcal{I}} \in (\leq nS.C)^{\mathcal{I}}$. Si la $\leq$-règle est applicable alors il y a au moins $n+1$ S-voisins $y_0, \ldots, y_n$ de x tels que $X_i \subseteq L(y_i)$ pour certains $X_i \in \mathsf{Flat}(C)$. De plus, il existe deux nœuds $y, z \in \{y_0, \ldots, y_n\}$ tels que $(\pi(y))^{\mathcal{I}} = (\pi(z))^{\mathcal{I}}$ (sinon nous aurions $(\pi(x))^{\mathcal{I}} \in (\geq nS.C)^{\mathcal{I}}$ vrai). $(\pi(y))^{\mathcal{I}} = (\pi(z))^{\mathcal{I}}$ implique que $y \neq z$ ne peut pas être satisfait à cause de (*) et y, z peuvent être choisis de telle sorte que y ne soit aucun nœud racine ni aucun ancêtre de z. Alors la $\leq$-règle peut être appliquée sans violer (*).

• La sat-règle : si la sat-règle n'est jamais appliquée sur x alors cette règle peut être appliquée sans violer (*) en choisissant $S = \bar{S} = L(\pi(x))$.

Cette affirmation produit la complétude du nouvel algorithme de tableau. Nous pouvons initier un arbre de complétion contenant un seul nœud $\hat{x}$ avec $L(\hat{x}) = \emptyset$ et $\neq = \emptyset$ et nous avons une fonction π satisfaisant (*) en fixant $\pi(\hat{x}) = s_0$ pour $s_0 \in \Delta$. Chaque fois qu'une règle est applicable à T, elle peut être appliquée de manière à maintenir

(*) et, selon le Lemme 2, toute séquence d'application de règles doit se terminer. Comme (*) est respecté, tout arbre généré par ces applications de règles doit être non-clash. Cela peut être considéré comme deux possibilités de clash à savoir :

— T ne peut pas contenir un nœud x tel que $\{C, \neg C\} \in L(x)$ car $L(x) \subseteq L(\pi(x))$ et donc $(\pi(x))^{\mathcal{I}} \in C^{\mathcal{I}} \cap (\neg C)^{\mathcal{I}}$ va être violé pour la définition de $C^{\mathcal{I}}$ et $(\neg C)^{\mathcal{I}}$.

— T ne peut pas contenir un nœud x avec $(\leq nSC) \in L(x)$ et $n+1$ S-voisins $y_0, \ldots, y_n$ de x avec $X \subseteq L(y_i)$ et $y_i \not\equiv y_j$ pour certains $X \in \mathsf{Flat}(C)$ et $0 \leq i < j \leq n$ car $(\leq nSC) \in L(\pi(x))$, et, puisque $y_i \not\equiv y_j$ implique $(\pi(y_i))^{\mathcal{I}} \neq (\pi(y_j))^{\mathcal{I}}$, $(\pi(x))^{\mathcal{I}} \in (\leq nSC)^{\mathcal{I}}$ est satisfait ; ce qui contredit la définition de $(\leq nSC)^{\mathcal{I}}$.

Donc, une fonction π satisfaisant (*) peut toujours être maintenue pour un arbre T ; c'est-à-dire qu'un arbre de complétion non-clash et complet peut être construit par l'application de l'Algorithme 1 à l'ontologie $\mathcal{O}$ cohérente.

□

Théorème 2. *L'opération de révision* $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$ *décrite dans la définition 6 satisfait les postulats* ***(D1)*****-*****(D6)***.

Démonstration. Démontrons que chaque postulat est satisfait par notre opération de révision.

[**(D1)**] Soient $\alpha = (C \sqsubseteq D)$ et $T \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')$. Par la Définition 6 (opération de révision), nous avons $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}')$. Ceci implique que $T \in \mathsf{MT}(\mathcal{O}')$ et $\mathcal{I}(T)$ est un modèle de $\mathcal{O}'$. Par conséquent, $C^{\mathcal{I}(T)} \sqsubseteq D^{\mathcal{I}(T)}$, et donc $\mathcal{I}(T) \models \alpha$.

[**(D2)**] Soit $T \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$. Prouvons que $T \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')$. Par la Définition 5 (distance), nous avons $T \vartriangle T = 0$. Car $T \vartriangle T = 0$, $T \triangle T \leq T_i \triangle T_j$ pour tous les modèles d'arbre $T_i \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ et $T_j \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$. De plus, par la Définition 6 (opération de révision), nous avons $T \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')$ (car $T \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ et $T \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$). Par conséquent, $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \subseteq \mathsf{MT}(\mathcal{O}, \mathcal{O}')$ (*).

Montrons que $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$. Soit $T_0 \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')$.

Par la Définition 6 (opération de révision), nous avons $T_0 \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$. Supposons que $T_0 \notin \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$. Par conséquent, T_0 est différent de chaque $T' \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$.

Par la Définition 5 (distance), $T_0 \vartriangle T' \neq 0$ pour chaque $T' \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$. De plus, comme $\mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \cap \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \neq \emptyset$, il existe un $T_1 \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \cap \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ avec $T_1 \vartriangle T_1 = 0$, et $T_1 \vartriangle T_1 < T_0 \vartriangle T'$ (car $T_0 \vartriangle T' \neq 0$). Par la Définition 6 (opération de révision), nous avons $T_0 \notin \mathsf{MT}(\mathcal{O}, \mathcal{O}')$, ce qui contredit $T_0 \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')$. Par conséquent, $T_0 \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ et $T_0 \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$. Donc, $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ (**). Enfin, par (*) & (**), nous avons $\mathsf{MT}(\mathcal{O}, \mathcal{O}') = \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$.

[**(D3)**] Supposons que $\mathsf{MT}(\mathcal{O}, \mathcal{O}') = \emptyset$. Comme $\mathcal{O}'$ est cohérente, $\mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \neq \emptyset$. Pour chaque $T'_0 \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ et $T_0 \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$, il existe toujours deux modèles d'arbre $T'_i \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ et $T_i \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ tels que $T'_i \vartriangle T_i < T'_0 \vartriangle T_0$ (car $\mathsf{MT}(\mathcal{O}, \mathcal{O}') = \emptyset$). Pour chaque paire de T'_i et T_i, nous pouvons toujours déterminer une autre paire de $T'_j \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ et $T_j \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ avec $i \neq j$ telle que $T'_j \vartriangle T_j < T'_i \vartriangle T_i$.

Cependant, le nombre de modèles d'arbre dans $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ et $\mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ est fini. Donc, il existe une paire $T'_j \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ et $T_j \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ telle qu'il n'existe pas d'autre paire $T'_n \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ et $T_n \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ pour assurer que $T'_n \vartriangle T_n < T'_j \vartriangle T_j$. Cela veut dire que $T'_j \vartriangle T_j \leq T'_n \vartriangle T_n$ pour tout $T'_n \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ et $T_n \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$. Donc, $T'_j \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')$ par la Définition 6 (opération de révision). Par conséquent, $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \neq \emptyset$.

[**(D4)**] Soit $T' \in \mathsf{MT}(\mathcal{O}_1, \mathcal{O}'_1)$. Par la Définition 6 (opération de révision), nous avons que $T' \in \mathsf{MT}(\mathcal{O}'_1, \mathsf{sub}(\mathcal{O}_1))$ et il existe $T \in \mathsf{MT}(\mathcal{O}_1, \mathsf{sub}(\mathcal{O}'_1))$ tel que $T' \vartriangle T \leq T_i \vartriangle T_j$ pour tout $T_i \in \mathsf{MT}(\mathcal{O}'_1, \mathsf{sub}(\mathcal{O}_1))$ et $T_j \in \mathsf{MT}(\mathcal{O}_1, \mathsf{sub}(\mathcal{O}'_1))$. Selon l'hypothèse, nous pouvons remplacer $\mathsf{MT}(\mathcal{O}'_1, \mathsf{sub}(\mathcal{O}_1))$ avec $\mathsf{MT}(\mathcal{O}'_2, \mathsf{sub}(\mathcal{O}_2))$ et $\mathsf{MT}(\mathcal{O}_1, \mathsf{sub}(\mathcal{O}'_1))$ avec $\mathsf{MT}(\mathcal{O}_2, \mathsf{sub}(\mathcal{O}'_2))$. Par conséquent, nous avons que $T' \in \mathsf{MT}(\mathcal{O}'_2, \mathsf{sub}(\mathcal{O}_2))$ et il existe $T \in \mathsf{MT}(\mathcal{O}_2, \mathsf{sub}(\mathcal{O}'_2))$ tel que $T' \vartriangle T \leq T_i \vartriangle T_j$ pour tout $T_i \in \mathsf{MT}(\mathcal{O}'_2, \mathsf{sub}(\mathcal{O}_2))$ et $T_j \in \mathsf{MT}(\mathcal{O}_2, \mathsf{sub}(\mathcal{O}'_2))$. Ceci implique que $T' \in \mathsf{MT}(\mathcal{O}_2, \mathcal{O}'_2)$. Donc, $\mathsf{MT}(\mathcal{O}_1, \mathcal{O}'_1) \subseteq \mathsf{MT}(\mathcal{O}_2, \mathcal{O}'_2)$.

Soit $T' \in \mathsf{MT}(\mathcal{O}_2, \mathcal{O}'_2)$. Par la Définition 6 (opération de révision), nous avons que $T' \in \mathsf{MT}(\mathcal{O}'_2, \mathsf{sub}(\mathcal{O}_2))$ et il existe $T \in \mathsf{MT}(\mathcal{O}_2, \mathsf{sub}(\mathcal{O}'_2))$ tel que $T' \vartriangle T \leq T_i \vartriangle T_j$ pour tous $T_i \in \mathsf{MT}(\mathcal{O}'_2, \mathsf{sub})$ et $T_j \in \mathsf{MT}(\mathcal{O}_2, \mathsf{sub}(\mathcal{O}'_2))$. Selon l'hypothèse, nous pouvons remplacer

$\mathsf{MT}(\mathcal{O}'_2, \mathsf{sub}(\mathcal{O}_2))$ avec $\mathsf{MT}(\mathcal{O}'_1, \mathsf{sub}(\mathcal{O}_1))$ et $\mathsf{MT}(\mathcal{O}_2, \mathsf{sub}(\mathcal{O}'_2))$ avec $\mathsf{MT}(\mathcal{O}_1, \mathsf{sub}(\mathcal{O}'_1))$. Par conséquent, nous avons que $T' \in \mathsf{MT}(\mathcal{O}'_1, \mathsf{sub}(\mathcal{O}_1))$ et il existe $T \in \mathsf{MT}(\mathcal{O}_1, \mathsf{sub}(\mathcal{O}'_1))$ tel que $T' \vartriangle T \leq T_i \vartriangle T_j$ pour tout $T_i \in \mathsf{MT}(\mathcal{O}'_1, \mathsf{sub}(\mathcal{O}_1))$ et $T_j \in \mathsf{MT}(\mathcal{O}_1, \mathsf{sub}(\mathcal{O}'_1))$. Ceci implique que $T' \in \mathsf{MT}(\mathcal{O}_1, \mathcal{O}'_1)$. Donc, $\mathsf{MT}(\mathcal{O}_2, \mathcal{O}'_2) \subseteq \mathsf{MT}(\mathcal{O}_1, \mathcal{O}'_1)$. Alors, $\mathsf{MT}(\mathcal{O}_1, \mathcal{O}'_1) = \mathsf{MT}(\mathcal{O}_2, \mathcal{O}'_2)$.

[**(D5)**] Soit $T \in \mathsf{MT}(\mathcal{O}, \mathcal{O}') \cap \mathsf{MT}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}'))$.
Car $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \cap \mathsf{MT}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')) \neq \emptyset$, un ensemble sub incluant $\mathsf{sub}(\mathcal{O})$, $\mathsf{sub}(\mathcal{O}')$ et $\mathsf{sub}(\mathcal{O}'')$ est utilisé lors de l'application de la sat-règle pour construire les modèles d'arbre pour les trois ontologies. Donc, $T \in \mathsf{MT}(\mathcal{O}, \mathcal{O}') \cap \mathsf{MT}(\mathcal{O}'', \mathsf{sub})$. Considérons les cas suivants :

- $\mathsf{MT}(\mathcal{O}, \mathsf{sub}) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}) \neq \emptyset$. Par **(D2)**, $T \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')$ implique $T \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub})$. De plus, $T \in \mathsf{MT}(\mathcal{O}'', \mathsf{sub})$ et donc $T \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}) \cap \mathsf{MT}(\mathcal{O}'', \mathsf{sub})$. Ainsi, $T \in \mathsf{MT}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub})$. Tout cela implique que $T \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}) \cap \mathsf{MT}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub})$ et donc $\mathsf{MT}(\mathcal{O}, \mathsf{sub}) \cap \mathsf{MT}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub}) \neq \emptyset$. Par **(D2)**, $T \in \mathsf{MT}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'')$.
- $\mathsf{MT}(\mathcal{O}, \mathsf{sub}) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}) = \emptyset$. Par la Définition 6 (opération de révision), nous avons $T \in \mathsf{MT}(\mathcal{O}', \mathsf{sub})$ (car $T \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')$). De plus, nous avons $T \in \mathsf{MT}(\mathcal{O}'', \mathsf{sub})$. Par conséquent, $T \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}) \cap \mathsf{MT}(\mathcal{O}'', \mathsf{sub}) = \mathsf{MT}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub})$. Supposons que $T \notin \mathsf{MT}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'')$. Par la Définition 6 (opération de révision), il existe $T_i \in \mathsf{MT}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub})$ et $T_j \in \mathsf{MT}(\mathcal{O}, \mathsf{sub})$ tels que $T_i \vartriangle T_j < T \vartriangle T'$ pour chaque $T' \in \mathsf{MT}(\mathcal{O}, \mathsf{sub})$ (car $T \notin \mathsf{MT}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'')$). En outre, $T_i \in \mathsf{MT}(\mathcal{O}', \mathsf{sub})$ car $T_i \in \mathsf{MT}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub}) = \mathsf{MT}(\mathcal{O}', \mathsf{sub}) \cap \mathsf{MT}(\mathcal{O}'', \mathsf{sub})$. Tout cela implique que $T_i \in \mathsf{MT}(\mathcal{O}', \mathsf{sub})$, $T_j \in \mathsf{MT}(\mathcal{O}, \mathsf{sub})$ et $T_i \vartriangle T_j < T \vartriangle T'$ pour chaque $T' \in \mathsf{MT}(\mathcal{O}, \mathsf{sub})$. Par la Définition 6 (opération de révision), $T \notin \mathsf{MT}(\mathcal{O}, \mathcal{O}')$, ce qui contredit $T \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')$. Donc, $T \in \mathsf{MT}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'')$.

Dans tous les cas, nous avons $T \in \mathsf{MT}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'')$. Donc, $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \cap \mathsf{MT}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')) \subseteq \mathsf{MT}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'')$.

[**(D6)**] Comme $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \cap \mathsf{MT}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')) \neq \emptyset$, un ensemble sub incluant $\mathsf{sub}(\mathcal{O})$, $\mathsf{sub}(\mathcal{O}')$ et $\mathsf{sub}(\mathcal{O}'')$ est utilisé lors de l'application de la sat-règle pour construire les modèles d'arbre des ontologies. Soit $T_1 \in \mathsf{MT}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'')$. Par la Définition 6 (opération de révision), il existe $T'_1 \in \mathsf{MT}(\mathcal{O}, \mathsf{sub})$ tel que $T_1 \vartriangle T'_1 \leq T_i \vartriangle T_j$ pour tout $T_i \in \mathsf{MT}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub})$ et $T_j \in \mathsf{MT}(\mathcal{O}, \mathsf{sub})$ (*). Nous pouvons trouver que $T_1 \in \mathsf{MT}(\mathcal{O}'', \mathsf{sub})$

ou $T_1 \in \mathsf{MT}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}'))$ (car $T_1 \in \mathsf{MT}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'') \subseteq \mathsf{MT}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub}(\mathcal{O})) = \mathsf{MT}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub}) = \mathsf{MT}(\mathcal{O}', \mathsf{sub}) \cap \mathsf{MT}(\mathcal{O}'', \mathsf{sub})$). Montrons que $T_1 \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')$. Comme $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \cap \mathsf{MT}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')) \neq \emptyset$, il existe $T_0 \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}) \cap \mathsf{MT}(\mathcal{O}'', \mathsf{sub})$ et $T_0' \in \mathsf{MT}(\mathcal{O}, \mathsf{sub})$ tels que $T_0 \vartriangle T_0' \leq T_i \vartriangle T_j$ pour tout $T_i \in \mathsf{MT}(\mathcal{O}', \mathsf{sub})$ et $T_j \in \mathsf{MT}(\mathcal{O}, \mathsf{sub})$ (**). Selon (*) et (**), nous avons $T_1 \vartriangle T_1' \leq T_0 \vartriangle T_0'$ (car $T_0 \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}) \cap \mathsf{MT}(\mathcal{O}'', \mathsf{sub}) = \mathsf{MT}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub})$) (***). Supposons que $T_1 \notin \mathsf{MT}(\mathcal{O}, \mathcal{O}')$. Ceci implique qu'il existe $T_2 \in \mathsf{MT}(\mathcal{O}', \mathsf{sub})$ et $T_2' \in \mathsf{MT}(\mathcal{O}, \mathsf{sub})$ tels que $T_2 \vartriangle T_2' < T_1 \vartriangle T_1'$ et $T_0 \vartriangle T_0' \leq T_2 \vartriangle T_2'$, ce qui contredit (***). □

Théorème 3. *Soient $\mathcal{O}$ et $\mathcal{O}'$, deux ontologies cohérentes en $\mathcal{SHIQ}$. L'ontologie de révision $\mathcal{O}^*$ de $\mathcal{O}$ par $\mathcal{O}'$ est une approximation maximale de $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$. En outre, la taille de $\mathcal{O}^*$ est bornée par une fonction doublement exponentielle de la taille de $\mathcal{O}$ et $\mathcal{O}'$.*

Démonstration. Pour démontrer que $\mathcal{O}^*$ est une approximation maximale de $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$, démontrons que $\mathcal{O}^*$ satisfait les trois conditions de la Définition 7 (approximation maximale) :

1. Par la construction de l'ontologie $\mathcal{O}^*$, cette ontologie contient tous les axiomes de l'ontologie $\mathcal{O}'$ et un nouvel axiome construit de $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$. Ceci implique que cet axiome est construit à partir des symboles se trouvant dans les ontologies $\mathcal{O}$ et $\mathcal{O}'$. Donc, $\mathcal{S}(\mathcal{O}^*) \subseteq \mathcal{S}(\mathcal{O}) \cup \mathcal{S}(\mathcal{O}')$.

2. Prouvons que $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}^*)$.

   Tout d'abord, montrons que $\mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}') = \mathsf{sub}(\mathcal{O}^*)$. Soit $C \in \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$. Par le comportement de la sat-règle, nous avons $C \in L(x)$ ou $\neg C \in L(x)$ pour chaque nœud x de chaque $T \in \mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$. Par la construction de $\mathcal{O}^*$, cette ontologie contient un nouvel axiome construit de $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$. Donc, C ou $\neg C$ doit se trouver dans chaque élément $(\bigsqcap_{C \in L_i(x)} C)$ de la disjonction $\bigsqcup_{1 \leq i \leq n, x \in V_i} (\bigsqcap_{C \in L_i(x)} C)$. Ceci implique que $\{C, \neg C\} \subseteq \mathsf{sub}(\mathcal{O}^*)$. Par conséquent, $\mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}') \subseteq \mathsf{sub}(\mathcal{O}^*)$. Inversement, par la définition de $\mathsf{sub}(\mathcal{O}^*)$ et la construction de $\mathcal{O}^*$, nous pouvons prouver que $\mathsf{sub}(\mathcal{O}^*) \subseteq \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$ de la même façon que précédemment. En effet, soit $C \in \mathsf{sub}(\mathcal{O}^*)$, nous avons aussi $\neg C \in \mathsf{sub}(\mathcal{O}^*)$ selon la définition de $\mathsf{sub}(\mathcal{O}^*)$. De plus, par la construction de $\mathcal{O}^*$, cette ontologie contient un nouvel

axiome construit de $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$. Ceci implique que C et $\neg C$ doivent apparaître dans les étiquettes des nœuds de tous les arbres dans $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$. Par conséquent, $\{C, \neg C\} \subseteq \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$ et donc $\mathsf{sub}(\mathcal{O}^*) \subseteq \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$. Alors, $\mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}') = \mathsf{sub}(\mathcal{O}^*)$.

Ensuite, nous montrons que $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}^*)$. Soit $T \in \mathsf{MT}(\mathcal{O}, \mathcal{O}')$. Par la définition de $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$, il existe une séquence Seq_T d'application de règles exécutée en appliquant le nouvel algorithme de tableau à $\mathcal{O}'$ avec l'importation de concepts $\mathsf{sub}(\mathcal{O})$ telle que Seq_T permet de construire $T = \langle V, L, E, \widehat{x} \rangle$. Pour montrer $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}^*)$, nous construisons une séquence $Seq_{T'}$ d'application de règles qui est exécutée en appliquant le nouvel algorithme de tableau à $\mathcal{O}^*$ telle que $Seq_{T'}$ permet de construire $T' = T$ avec $T' = \langle V', L', E', \widehat{x}' \rangle$.

Cas de base. En appliquant le nouvel algorithme de tableau à $\mathcal{O}'$ avec l'importation de concepts $\mathsf{sub}(\mathcal{O})$, la première règle R_1 dans Seq_T doit être une application de la sat-règle qui ajoute dans l'étiquette du nœud racine $\widehat{x}$ deux ensembles S et $\bar{S}$ tels que $S \subseteq \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$ avec $X \subseteq S$ pour certains $X \in \mathsf{Flat}(\neg C \sqcup D)$ et pour chaque axiome $C \sqsubseteq D \in \mathcal{O}'$. Soit T_{R_1} le modèle d'arbre intermédiaire construit par l'application de la règle R_1 dans Seq_T. Dans ce cas, nous ajoutons dans $Seq_{T'}$ la première règle R'_1 de la sat-règle qui choisit les mêmes ensembles S et $\bar{S}$ car $\mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}') = \mathsf{sub}(\mathcal{O}^*)$. Soit $T'_{R'_1}$, le modèle d'arbre intermédiaire construit par l'application de la règle R'_1 dans $Seq_{T'}$. Un isomorphisme π entre V' et V est initié tel que $\pi(\widehat{x}') = \widehat{x}$, et deux modèles d'arbre T_{R_1} et $T'_{R'_1}$ sont équivalents.

Cas général. Soit T_{R_i} le modèle d'arbre intermédiaire construit par l'application des règles $R_1, \cdots, R_i$ où $Seq_T = \{R_1, \cdots, R_i, R_{i+1}, \cdots, R_k\}$, soit $x \in V$, le nœud qui est traité par R_i, soit $T'_{R'_i}$, le modèle d'arbre intermédiaire construit par l'application des règles $R'_1, \cdots, R'_i$ où $Seq_{T'} = \{R'_1, \ldots, R'_i, R'_{i+1}, \cdots, R'_k\}$, soit $x' \in V'$, le nœud traité par R'_i tel que $\pi(x') = x$ et deux modèles d'arbre T_{R_i} et $T'_{R'_i}$ sont équivalents. Considérons les cas suivants pour la règle R_{i+1} dans Seq_T et montrons que l'isomorphisme π est toujours maintenu et que deux modèles d'arbre $T_{R_{i+1}}$ et $T'_{R'_{i+1}}$ sont équivalents.

- R_{i+1} est une sat-règle. En appliquant le nouvel algorithme de tableau à $\mathcal{O}'$ avec

l'importation de concepts $\mathsf{sub}(\mathcal{O})$, la règle R_{i+1} ajoute dans l'étiquette d'un nœud y de T_{R_i}, deux ensembles S et $\bar{S}$ tels que $S \subseteq \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$. Dans ce cas, nous ajoutons dans $Seq_{T'}$ une application R'_{i+1} de la sat-règle qui choisit les mêmes ensembles S et $\bar{S}$ (car $\mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}') = \mathsf{sub}(\mathcal{O}^*)$) pour ajouter dans l'étiquette du nœud y' de $T_{R'_i}$ avec $\pi(y') = y$. Donc, l'isomorphisme π est maintenu.

• R_{i+1} est une $\forall$-règle. Cela implique que R_{i+1} est applicable à un concept $\forall S.C$ dans l'étiquette de $\pi(y') = y$ qui appartient à T_{R_i}. Soit $\pi(z') = z$ un S-voisin de $\pi(y')$ tel que R_{i+1} ajoute un sous-ensemble $X \in \mathsf{Flat}(C)$ à $L(\pi(z'))$. Comme π assure l'équivalence entre T_{R_i} et $T'_{R'_i}$, R_{i+1} est aussi applicable au concept $\forall S.C$ dans l'étiquette de y' appartenant à $T'_{R'_i}$ tel que R_{i+1} ajoute un sous-ensemble $X \in \mathsf{Flat}(C)$ dans $L'(z')$ sur $T'_{R'_i}$. $R'_{i+1} = R_{i+1}$ est ajouté dans $Seq_{T'}$. Alors, π est maintenu.

• R_{i+1} est une $\forall_+$-règle. Cela implique que R_{i+1} est applicable à un concept $\forall S.C$ dans l'étiquette de $\pi(y') = y$ appartenant à T_{R_i}. Soient R, un rôle avec $R \boxplus S$ et $\pi(z') = z$, un S-voisin de $\pi(y')$ tels que R_{i+1} ajoute $\forall R.C$ dans $L(\pi(z'))$. Comme π assure l'équivalence entre T_{R_i} et $T'_{R'_i}$, R_{i+1} est aussi applicable au concept $\forall S.C$ dans l'étiquette de y' appartenant à $T'_{R'_i}$ tel que R_{i+1} ajoute $\forall R.C$ dans $L'(z')$ sur $T'_{R'_i}$. $R'_{i+1} = R_{i+1}$ est ajouté dans $Seq_{T'}$. Alors, π est maintenu.

• R_{i+1} est une $\exists$- ou $\geq$-règle. Ceci implique que R_{i+1} est applicable sur $\pi(y') = y$ appartenant à T_{R_i}. Soit $y_1, \cdots, y_m$, les nœuds créés par R_{i+1} dans $T_{R_{i+1}}$. Comme π assure l'équivalence entre T_{R_i} et $T'_{R'_i}$, R_{i+1} est aussi applicable sur y' appartenant à $T'_{R'_i}$. Soit $y'_1, \cdots, y'_m$, les nœuds créés par R'_{i+1} dans $T'_{R'_{i+1}}$. $R'_{i+1} = R_{i+1}$ est ajouté dans $Seq_{T'}$. π est étendu de telle façon que $\pi(y'_i) = y_i$ pour $1 \leq i \leq m$. Il est évident que π est maintenu.

• R_{i+1} est une $\leq$-règle. Ceci implique que R_{i+1} est applicable sur $\pi(y') = y$ appartenant à T_{R_i}. Soit $\pi(y'_0) = y_0$, le nœud fusionné à un voisin $\pi(z') = z$ de $\pi(y') = y$ par R_{i+1} dans $T_{R_{i+1}}$. Comme π assure l'équivalence entre T_{R_i} et $T'_{R'_i}$, R_{i+1} est aussi applicable sur y' appartenant à $T'_{R'_i}$ et fusionne le nœud y'_0 au voisin z' de y'. $R'_{i+1} = R_{i+1}$ est ajouté dans $Seq_{T'}$. Il est évident que π est maintenu.

Dans tous les cas de R_{i+1}, l'isomorphisme π est maintenu et deux modèles d'arbre $T_{R_{i+1}}$ et $T'_{R'_{i+1}}$ sont équivalents. Puis, nous construisons $T' = T$ en utilisant la sé-

quence $Seq_{T'}$. Donc, $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}^*)$.

3. Prouvons la troisième condition : Il n'existe aucune ontologie $\mathcal{O}''$ telle que $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}'') \subset \mathsf{MT}(\mathcal{O}^*)$. Pour ce faire, on suppose qu'il existe une ontologie $\mathcal{O}''$ telle que $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}'') \subset \mathsf{MT}(\mathcal{O}^*)$. Démontrons que $\mathsf{MT}(\mathcal{O}'') = \mathsf{MT}(\mathcal{O}^*)$.

   D'abord, montrons que $\mathsf{sub}(\mathcal{O}'') = \mathsf{sub}(\mathcal{O}^*)$. Soient $T' = \langle V', L', E', \widehat{x}' \rangle \in \mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}'') \subset \mathsf{MT}(\mathcal{O}^*)$ et $x' \in V'$.

   Soit $C \in \mathsf{sub}(\mathcal{O}'')$, nous avons aussi $\neg C \in \mathsf{sub}(\mathcal{O}'')$ selon la définition de $\mathsf{sub}(\mathcal{O}'')$. Par le comportement de la sat-règle, nous avons $C' \in L'(x')$ où $C' \in \{C, \neg C\}$ (car $C, \neg C \in \mathsf{sub}(\mathcal{O}'')$ et $T' \in \mathsf{MT}(\mathcal{O}'')$). De même, par le comportement de la sat-règle, nous avons $C' \in \mathsf{sub}(\mathcal{O}^*)$ (car $C' \in L'(x')$ et $T' \in \mathsf{MT}(\mathcal{O}^*)$), ceci implique $C \in \mathsf{sub}(\mathcal{O}^*)$. Par conséquent, $\mathsf{sub}(\mathcal{O}'') \subseteq \mathsf{sub}(\mathcal{O}^*)$.

   Inversement, soit $C \in \mathsf{sub}(\mathcal{O}^*)$, nous avons aussi $\neg C \in \mathsf{sub}(\mathcal{O}^*)$ selon la définition de $\mathsf{sub}(\mathcal{O}^*)$. Par le comportement de la sat-règle, nous avons $C' \in L(x)$ où $C' \in \{C, \neg C\}$ pour chaque nœud x de chaque modèle d'arbre $T \in \mathsf{MT}(\mathcal{O}^*)$ (car $C, \neg C \in \mathsf{sub}(\mathcal{O}^*)$). Puisque nous avons déjà $T' \in \mathsf{MT}(\mathcal{O}^*)$ alors $C' \in L'(x')$. Par le comportement de la sat-règle, nous avons $C' \in \mathsf{sub}(\mathcal{O}'')$ (car $C' \in L'(x')$ et $T' \in \mathsf{MT}(\mathcal{O}'')$), cela implique $C \in \mathsf{sub}(\mathcal{O}'')$. Par conséquent, $\mathsf{sub}(\mathcal{O}^*) \subseteq \mathsf{sub}(\mathcal{O}'')$. Donc, $\mathsf{sub}(\mathcal{O}'') = \mathsf{sub}(\mathcal{O}^*)$.

   Montrons maintenant que s'il existe $\mathcal{O}''$ telle que $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}'') \subseteq \mathsf{MT}(\mathcal{O}^*)$ alors $\mathsf{MT}(\mathcal{O}'') = \mathsf{MT}(\mathcal{O}^*)$. Soit $T \in \mathsf{MT}(\mathcal{O}^*)$, ceci implique qu'il existe une séquence Seq_T d'application de règles exécutée par le nouvel algorithme de tableau appliquant à $\mathcal{O}^*$ telle que Seq_T permet de construire $T = \langle V, L, E, \widehat{x} \rangle$.

   Pour prouver $\mathsf{MT}(\mathcal{O}'') = \mathsf{MT}(\mathcal{O}^*)$, nous construisons une séquence $Seq_{T'}$ d'application de règles exécutée par le nouvel algorithme de tableau appliquant à $\mathcal{O}''$ telle que $Seq_{T'}$ permet de construire un modèle d'arbre $T' = T$ avec $T' = \langle V', L', E', \widehat{x}' \rangle$.

   Cas de base. Par le nouvel algorithme de tableau appliqué à $\mathcal{O}^*$, la première règle R_1 dans Seq_T doit être une application de la sat-règle qui ajoute dans l'étiquette du nœud racine $\widehat{x}$ deux ensembles S et $\bar{S}$ tels que $S \subseteq \mathsf{sub}(\mathcal{O}^*)$ et $X \subseteq S$ pour $X \in \mathsf{Flat}(\neg C \sqcup D)$ et pour chaque axiome $C \sqsubseteq D \in \mathcal{O}^*$. Soit T_{R_1}, le modèle d'arbre intermédiaire construit par l'application de la règle R_1 dans Seq_T. Dans ce cas, nous ajoutons dans $Seq_{T'}$ la première règle R'_1 de la sat-règle qui choisit les mêmes

ensembles S et $\bar{S}$ car $\mathsf{sub}(\mathcal{O}'') = \mathsf{sub}(\mathcal{O}^*)$. Soit $T'_{R'_1}$, le modèle d'arbre intermédiaire construit par l'application de la règle R'_1 dans $Seq_{T'}$, un isomorphisme est initié π entre V' et V tel que $\pi(\widehat{x}') = \widehat{x}$ et deux modèles d'arbre T_{R_1} et $T'_{R'_1}$ sont équivalents.

Cas général. Soit T_{R_i}, le modèle d'arbre intermédiaire construit par l'application des règles $R_1, \cdots, R_i$ où $Seq_T = \{R_1, \cdots, R_i, R_{i+1}, \cdots, R_k\}$, soit $x \in V$, le nœud qui est traité par R_i et $T'_{R'_i}$, le modèle d'arbre intermédiaire construit par l'application de la règle $R'_1, \cdots, R'_i$ où $Seq_{T'} = \{R'_1, \ldots, R'_i, R'_{i+1}, \ldots, R'_k\}$, soit $x' \in V'$, le nœud qui est traité par R'_i tel que $\pi(x') = x$, et deux modèles d'arbre T_{R_i} et $T'_{R'_i}$ équivalents.

Considérons les cas suivants pour la règle R_{i+1} dans Seq_T.

• R_{i+1} est une sat-règle. Par le nouvel algorithme de tableau appliquant à $\mathcal{O}^*$, la règle R_{i+1} ajoute dans l'étiquette d'un nœud y de T_{R_i} deux ensembles S et $\bar{S}$ tels que $S \subseteq \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$ et $X \subseteq S$ pour $X \in \mathsf{Flat}(\neg C \sqcup D)$ et pour chaque axiome $C \sqsubseteq D \in \mathcal{O}^*$. Dans ce cas, nous ajoutons dans $Seq_{T'}$ une application R'_{i+1} de la sat-règle qui choisit les mêmes ensembles S et $\bar{S}$ car $\mathsf{sub}(\mathcal{O}'') = \mathsf{sub}(\mathcal{O}^*)$. L'isomorphisme π est maintenu.

• R_{i+1} est une $\forall$-règle. Ceci implique que R_{i+1} est applicable à un concept $\forall S.C$ dans l'étiquette de $\pi(y') = y$ qui appartient à T_{R_i}. Soit $\pi(z') = z$, un S-voisin de $\pi(y')$ tel que R_{i+1} ajoute un sous-ensemble $X \in \mathsf{Flat}(C)$ dans $L(\pi(z'))$. Comme π assure l'équivalence entre T_{R_i} et $T'_{R'_i}$, R_{i+1} est aussi applicable au concept $\forall S.C$ dans l'étiquette de y' appartenant à $T'_{R'_i}$ tel que R_{i+1} ajoute un sous-ensemble $X \in \mathsf{Flat}(C)$ dans $L'(z')$ sur $T'_{R'_i}$. $R'_{i+1} = R_{i+1}$ est ajouté dans $Seq_{T'}$. Alors, π est maintenu.

• R_{i+1} est une $\forall_+$-règle. Cela implique que R_{i+1} est applicable à un concept $\forall S.C$ dans l'étiquette de $\pi(y') = y$ qui appartient à T_{R_i}. Soient R, un rôle avec $R \sqsubseteq S$ et $\pi(z') = z$, un S-voisin de $\pi(y')$ tels que R_{i+1} ajoute $\forall R.C$ dans $L(\pi(z'))$. Comme π assure l'équivalence entre T_{R_i} et $T'_{R'_i}$, R_{i+1} est aussi applicable au concept $\forall S.C$ dans l'étiquette de y' appartenant à $T'_{R'_i}$ tel que R_{i+1} ajoute $\forall R.C$ dans $L'(z')$ sur $T'_{R'_i}$. $R'_{i+1} = R_{i+1}$ est ajouté dans $Seq_{T'}$. Alors, π est maintenu.

• R_{i+1} est une $\exists$- ou $\geq$-règle. Ceci implique que R_{i+1} est applicable sur $\pi(y') = y$ qui appartient à T_{R_i}. Soient $y_1, \cdots, y_m$, les nœuds créés par R_{i+1} dans $T_{R_{i+1}}$. Comme π

assure l'équivalence entre T_{R_i} et $T'_{R'_i}$, R_{i+1} est aussi applicable sur y' appartenant à $T'_{R'_i}$. Soient $y'_1, \cdots, y'_m$, les nœuds créés par R'_{i+1} dans $T'_{R'_{i+1}}$, $R'_{i+1} = R_{i+1}$ est ajouté dans $Seq_{T'}$. π est étendu de telle façon que $\pi(y'_i) = y_i$ pour $1 \leq i \leq m$. Alors, π est maintenu.

• R_{i+1} est une $\leq$-règle. Ceci implique que R_{i+1} est applicable sur $\pi(y') = y$ qui appartient à T_{R_i}. Soit $\pi(y'_0) = y_0$, le nœud fusionné à un voisin $\pi(z') = z$ de $\pi(y') = y$ par R_{i+1} dans $T_{R_{i+1}}$. Comme π assure l'équivalence entre T_{R_i} et $T'_{R'_i}$, R_{i+1} est aussi applicable sur y' appartenant à $T'_{R'_i}$ et fusionne le nœud y'_0 au voisin z' de y'. $R'_{i+1} = R_{i+1}$ est ajouté dans $Seq_{T'}$. Alors, π est maintenu.

Dans tous les cas de R_{i+1}, l'isomorphisme π est maintenu et deux modèles d'arbre $T_{R_{i+1}}$ et $T'_{R'_{i+1}}$ sont équivalents. Donc, la séquence $Seq_{T'}$ d'applications de règle exécutée par le nouvel algorithme de tableau appliquant à $\mathcal{O}''$ construit un modèle d'arbre $T' = T$. Par conséquent, $\mathsf{MT}(\mathcal{O}^*) \subseteq \mathsf{MT}(\mathcal{O}'')$. Selon l'hypothèse, nous avons aussi $\mathsf{MT}(\mathcal{O}'') \subseteq \mathsf{MT}(\mathcal{O}^*)$. Alors, $\mathsf{MT}(\mathcal{O}'') = \mathsf{MT}(\mathcal{O}^*)$. Par conséquent, il n'existe aucune ontologie $\mathcal{O}''$ telle que $\mathsf{MT}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{MT}(\mathcal{O}'') \subset \mathsf{MT}(\mathcal{O}^*)$.

□

Lemme 4. ***(Terminaison, Correction et Complétude).*** *Soit $\mathcal{O}$ une ontologie en $\mathcal{SHIQ}$.*

1. *L'algorithme de tableau avec les règles d'expansion de la figure 4.3 se termine.*
2. *Si l'algorithme de tableau avec les règles d'expansion de la figure 4.3 peut être appliqué à $\mathcal{O}$ de telle façon qu'il construise un arbre de complétion non-clash et complet alors $\mathcal{O}$ est cohérente ;*
3. *Si $\mathcal{O}$ est cohérente alors l'algorithme de tableau avec les règles d'expansion de la figure 4.3 peut être appliqué à $\mathcal{O}$ de telle façon qu'il construise un arbre de complétion non-clash et complet.*

Démonstration.

1. Soient $m = |\mathsf{sub}(\mathcal{O})|/2$, $k = |\mathbf{R}|$. La terminaison de l'algorithme de tableau est une conséquence des propriétés suivantes :

(a) L'application des règles dans la Figure 4.3 ne supprime jamais les nœuds de la forêt ou des concepts à partir des étiquettes de nœud. Les étiquettes des arêtes ne peuvent être modifiées que par les $\leq$- et $\leq_r$-règles qui les étendent ou les mettent à $\emptyset$, et lors que une étiquette des arêtes devient vide par la $\leq$-règle alors les nœuds situés sous cette arête seront ignorés et ne seront jamais traités. La $\leq_r$-règle ne met que l'étiquette d'un nœud racine à $\emptyset$, et après cela, l'étiquette de ce nœud racine n'est jamais changé à nouveau puisque tous les arêtes à/de ce nœud racine sont supprimées. Comme aucun nœud racine est généré, cette suppression ne peut se produire qu'un nombre fini de fois, et les nouvelles arêtes générées par la $\leq_r$-règle garantissent que la structure résultante est toujours une forêt de complétion.

(b) Pour un nœud x, les successeurs de x doivent être le résultat d'une application de la $\exists$-règle ou la $\geq$-règle à des concepts de la forme $\exists R.C$ et $(\geq nS.C)$ dans $L(x)$. De plus, La génération de successeurs est effectuée au plus une fois de chacun de ces concepts.

(c) Les nœuds sont étiquetés par des sous-ensembles non vides de $\mathsf{sub}(\mathcal{O})$ et les arêtes par des sous-ensembles de $\mathbf{R}$, il y a donc au plus $K = 2^{2mk}$ différentes étiquettes possibles pour une paire de nœuds et une arête. Alors, si un chemin p est de longueur au moins K, La condition de blocage "paire-wise" implique l'existence de deux nœuds x, y sur p telle que x bloque y. Puisque un chemin sur lequel les nœuds sont bloqués ne peut pas devenir plus longueur, les chemins sont de longueur au plus K. Et le nombre de voisins de n'importe quel nœud est limité par $M = \sum m_i$ où m_i se trouve dans $\geq m_i S.C$ qui apparaît dans $\mathcal{O}$.

2. Soit $\mathcal{F} = (G, T\langle\widehat{x}_1\rangle, \cdots, T\langle\widehat{x}_n\rangle)$ une forêt de complétion non-clash et complète avec $G = (\mathbf{V}, \mathbf{E}, \mathbf{L})$ et $T\langle\widehat{x}_i\rangle = (V_i, E_i, L_i)$ $(1 \leq i \leq n)$. Un *chemin* est une séquence de paires de nœuds de $\mathcal{F}$ de la forme $p = [\frac{x_0}{x'_0}, \ldots, \frac{x_n}{x'_n}]$ qui commence d'un nœud racine à un autre qui est bloqué, et qui ne va que via des nœuds non-racine. Pour chaque chemin nous définissons $\mathsf{Tip}(p) := x_n$ et $\mathsf{Tip}'(p) := x'_n$. Avec $[p|\frac{x_{n+1}}{x'_{n+1}}]$, nous désignons le chemin $[\frac{x_0}{x'_0}, \ldots, \frac{x_n}{x'_n}, \frac{x_{n+1}}{x'_{n+1}}]$. L'ensemble $\mathsf{Paths}(\mathcal{F})$ est défini par induction comme suit :

• Pour le nœud racine $\widehat{x}_i$ de $\mathcal{F}$, $[\frac{\widehat{x}_i}{\widehat{x}_i}] \in \mathsf{Paths}(\mathcal{F})$, et

• Pour chaque chemin $p \in \mathsf{Paths}(\mathcal{F})$ et chaque z est un successeur de $\mathsf{Tip}(p)$ dans $\mathcal{F}$:

— si z n'est pas bloqué ni un nœud racine , alors $[p|\frac{z}{z}] \in \mathsf{Paths}(\mathcal{F})$, et

— si z est bloqué par un nœud y, alors $[p|\frac{y}{z}] \in \mathsf{Paths}(\mathcal{F})$.

Par la construction de $\mathsf{Paths}(\mathcal{F})$ et la condition de blocage, notons que

— si $p \in \mathsf{Paths}(\mathcal{F})$, alors $\mathsf{Tip}(p)$ n'est pas bloqué,

— $\mathsf{Tip}(p) = \mathsf{Tip}'(p)$ ssi $\mathsf{Tip}'(p)$ n'est pas bloqué, et

— $L(\mathsf{Tip}(p)) = L(\mathsf{Tip}'(p))$.

Nous définissons une interprétation $\mathcal{I} = (\Delta, .^{\mathcal{I}})$ comme suit :

• $\Delta := \mathsf{Paths}(\mathcal{F})$

• $A^{\mathcal{I}} := \{p \mid A \in L(\mathsf{Tip}(p))\}$ pour tout nom de concept A dans $\mathsf{sub}(O)$

• Pour chaque individu $a_i \in \mathbf{I}$,

$$a_i^{\mathcal{I}} := \begin{cases} [\frac{\widehat{x}_i}{\widehat{x}_i}] \text{ si } \widehat{x}_i \text{ est un nœud racine dans } \mathcal{F} \text{ avec } L(\widehat{x}_i) \neq \emptyset \\ [\frac{\widehat{x}_j}{\widehat{x}_j}] \text{ si } L(\widehat{x}_i) = \emptyset, \widehat{x}_j \text{ est un nœud racine dans } \mathcal{F} \text{ avec } L(\widehat{x}_j) \neq \emptyset \text{ et } \widehat{x}_i \doteq \widehat{x}_j \end{cases}$$

• $$R^{\mathcal{I}} := \begin{cases} \mathcal{E}(R)^+ \text{ si } \mathsf{Trans}(R) = \mathsf{vrai} \\ \mathcal{E}(R) \cup \bigcup_{P \sqsubseteq R, P \neq R} P^{\mathcal{I}} \text{ sinon} \end{cases}$$

où $\mathcal{E}(R)^+$ est la fermeture transitive de $\mathcal{E}(R) := \{\langle p, [p|\frac{x}{x'}]\rangle \in \Delta \times \Delta \mid x' \text{ est un } R\text{-successeur de } \mathsf{Tip}(p)\} \cup \{\langle [q|\frac{x}{x'}], q\rangle \in \Delta \times \Delta \mid x' \text{ est un } \mathsf{Inv}(R)\text{-successeur de } \mathsf{Tip}(q)\} \cup \{\langle [\frac{x}{x}], [\frac{y}{y}]\rangle \in \Delta \times \Delta \mid x, y \text{ sont des nœuds racines, et } y \text{ est un } R\text{-voisin de } x\}$. L'interprétation récursive des rôles non-transitives est utile pour interpréter correctement ces rôles non-transitives ayant un sous-rôle transitive.

Pour montrer que $\mathcal{I} = (\Delta, .^{\mathcal{I}})$ est un modèle de $\mathcal{O}$, nous devons démontrer que (i) $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ pour chaque $R \sqsubseteq S \in \mathcal{R}$, (ii) $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ pour chaque $C \sqsubseteq D \in \mathcal{T}$, (iii) $a^{\mathcal{I}} \in C^{\mathcal{I}}$ et $\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$ pour chaque $C(a) \in \mathcal{A}$ et $R(a, b) \in \mathcal{A}$ respectivement.

(i) $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ pour chaque $R \sqsubseteq S \in \mathcal{R}$. Supposons que $\mathsf{Trans}(R) = \mathsf{false}$. Selon l'application des règles sur $\mathcal{F}$, si $R \in L(\langle x, y\rangle)$ alors $S \in L(\langle x, y\rangle)$ où x, y sont

deux nœuds de $\mathcal{F}$. À cause de la définition de $R^{\mathcal{I}}$ et $S^{\mathcal{I}}$, nous avons $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$. Supposons que $\mathsf{Trans}(R) = \mathsf{true}$. Si $\mathsf{Trans}(S) = \mathsf{false}$, nous avons $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ en raison de la définition de $S^{\mathcal{I}}$. Sinon, $\mathsf{Trans}(S) = \mathsf{true}$. Puisque l'application des règles sur $\mathcal{F}$ impose que si $R \in L(\langle x, y\rangle)$ alors $S \in L(\langle x, y\rangle)$ où x, y sont deux nœuds de $\mathcal{F}$, nous avons aussi $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ en raison de la définition de $R^{\mathcal{I}}$ et $S^{\mathcal{I}}$.

(ii) $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ pour chaque $C \sqsubseteq D \in \mathcal{T}$. D'après la sat-règle appliquant sur chaque nœud x de $\mathcal{F}$, il existe toujours $X \in \mathsf{Flat}(\neg C \sqcup D)$ tel que $X \subseteq L(x)$. À cause de la définition de Flat, nous pouvons écrire $X = Y \cup Z$ où $Y \in \mathsf{Flat}(\neg C)$ et $Z \in \mathsf{Flat}(D)$. Let $\bar{X} = X_1 \sqcap \cdots \sqcap X_n$ où $X_i \in X$ $(1 \leq i \leq n)$, $\bar{Y} = Y_1 \sqcap \cdots \sqcap Y_m$ où $Y_j \in Y$ $(1 \leq j \leq m)$ et $\bar{Z} = Z_1 \sqcap \cdots \sqcap Z_k$ où $Z_t \in Z$ $(1 \leq t \leq k)$. Cela implique qu'il existe $p \in \bar{X}^{\mathcal{I}} = \bar{Y}^{\mathcal{I}} \cup \bar{Z}^{\mathcal{I}} \subseteq (\neg C)^{\mathcal{I}} \cup D^{\mathcal{I}}$ où $\mathsf{Tip}(p) = x$. Alors, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

(iii) $a^{\mathcal{I}} \in C^{\mathcal{I}}$ et $\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$ pour chaque $C(a) \in \mathcal{A}$ et $R(a, b) \in \mathcal{A}$ respectivement. D'après l'initialisation de $\mathcal{F}$, il existe toujours $X \in \mathsf{Flat}(C)$ et un nœud racine $\widehat{x}$ de $\mathcal{F}$ (resp. des nœuds racines $\widehat{x}$, $\widehat{y}$ de $\mathcal{F}$) tel que $X \subseteq L(x)$ et $\{a\} \in L(\widehat{x})$ (resp. $R \in L(\langle \widehat{x}, \widehat{y}\rangle)$, $\{a\} \in L(\widehat{x})$ et $\{b\} \in L(\widehat{y})$). Cela implique qu'il existe $p \in \bar{X}^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ (resp. $\langle p, q\rangle \in R^{\mathcal{I}}$) où $\mathsf{Tip}(p) = \widehat{x}$ (resp. $\mathsf{Tip}(p) = \widehat{x}$ et $\mathsf{Tip}(q) = \widehat{y}$) et $\bar{X} = X_1 \sqcap \cdots \sqcap X_n$ avec $X_i \in X$ $(1 \leq i \leq n)$. À cause de la définition de $a^{\mathcal{I}}$ et $b^{\mathcal{I}}$, nous avons $a^{\mathcal{I}} \in C^{\mathcal{I}}$ et $\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$.

3. Il existe un modèle $\mathcal{I} = \langle \Delta, .^{\mathcal{I}}\rangle$ tel que $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, $a^{\mathcal{I}} \in C^{\mathcal{I}}$ et $\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$ pour tout axiome et toute assertion $C \sqsubseteq D$, $C(a)$ et $R(a, b)$ dans $\mathcal{O}$. Cela signifie que tous les sous-concepts X de l'ontologie $\mathcal{O}$ doivent être interprétés par $\mathcal{I}$ (car sinon nous ne pouvons pas savoir si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ ou $a^{\mathcal{I}} \in C^{\mathcal{I}}$ avec X se trouve C ou D). Donc, X est interprété par $\mathcal{I}$ pour chaque $X \in \mathsf{sub}(\mathcal{O})$. Alors, le modèle $\mathcal{I}$ de $\mathcal{O}$ est aussi un modèle d'une ontologie $\mathcal{O}' = \mathcal{O} \cup \{X \sqsubseteq X \mid X \in \mathsf{sub}(\mathcal{O})\}$.

   Nous utilisons ce modèle pour traiter l'application des règles. Pour ce faire, nous allons définir par induction une fonction π, associant les nœuds de $\mathcal{F}$ à Δ comme suit :

   — Pour des individus a_i dans $\mathcal{A}$, nous définissons $\pi(\widehat{x}_i) = a_i^{\mathcal{I}}$.

   — Si $\pi(x) = s$ est déjà définie, et un successeur y de x est généré pour $\exists R.C \in$

$L(x)$, alors $\pi(y) = t$ pour un $t \in \Delta$ avec $X \subseteq L(t)$ et $\langle s, t\rangle \in R^{\mathcal{I}}$ pour un $X \in \mathsf{Flat}(C)$.

— Si $\pi(x) = s$ est déjà définie, et un successeur y de x est généré pour $(\geq nR.C) \in L(x)$, alors $\pi(y_i) = t_i$ pour n distincts $t_i \in \Delta$ avec $X \subseteq L(t_i)$ et $\langle s, t_i\rangle \in R^{\mathcal{I}}$ pour un $X \in \mathsf{Flat}(C)$.

Et l'association pour la forêt de complétion initiale $\mathcal{F}$ satisfait les conditions suivantes, pour chaque x, y dans $\mathcal{F}$:

$$\left.\begin{array}{l} L(x) \subseteq L(\pi(x)) = \{C \mid (\pi(x))^{\mathcal{I}} \in C^{\mathcal{I}}\} \\ \text{si } y \text{ est un S-voisin de } x \text{ alors } \langle(\pi(x))^{\mathcal{I}}, (\pi(y))^{\mathcal{I}}\rangle \in S^{\mathcal{I}} \\ x \not\doteq y \Rightarrow (\pi(x))^{\mathcal{I}} \neq (\pi(y))^{\mathcal{I}} \end{array}\right\} (*)$$

CLAIM : Soient $\mathcal{F}$ une forêt de complétion et π une fonction qui satisfait (*). Si une règle applicable à $\mathcal{F}$ alors la règle est applicable à $\mathcal{F}$ dans une manière qui donne une forêt de complétion $\mathcal{F}'$ et une fonction π' qui satisfait (*).

Soient $\mathcal{F}$ une forêt de complétion et π une fonction qui satisfait (*). Nous devons considérer les règles suivantes.

• La $\exists$-règle : Si $\exists S.C \in L(x)$, alors $\exists S.C \in L(\pi(x))$ et $(\pi(x))^{\mathcal{I}} \in (\exists S.C)^{\mathcal{I}}$. À cause de la définition de $(\exists S.C)^{\mathcal{I}}$, il y a un individu $t \in \Delta$ tel que $\langle(\pi(x))^{\mathcal{I}}, t^{\mathcal{I}}\rangle \in S^{\mathcal{I}}$ et $t^{\mathcal{I}} \in C^{\mathcal{I}}$. L'application de la $\exists$-règle génère une nouvelle variable y avec $L(\langle x, y\rangle) = \{S\}$ et $L(y) = X$ pour un $X \in \mathsf{Flat}(C)$ tel que $t^{\mathcal{I}} \in X^{\mathcal{I}} \subseteq C^{\mathcal{I}}$. Donc, nous mettons $\pi' := \pi[y \to t]$ qui donne une fonction qui satisfait (*) pour la forêt modifiée.

• La $\forall$-règle : Si $\forall S.C \in L(x)$, alors $\forall S.C \in L(\pi(x))$, et $(\pi(x))^{\mathcal{I}} \in (\forall S.C)^{\mathcal{I}}$. De plus, si y est un S-voisin de x, alors $\langle(\pi(x))^{\mathcal{I}}, (\pi(y))^{\mathcal{I}}\rangle \in S^{\mathcal{I}}$ en raison de (*). À cause de la définition de $(\forall S.C)^{\mathcal{I}}$, nous avons $(\pi(y))^{\mathcal{I}} \in X^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ pour un $X \in \mathsf{Flat}(C)$. Cela implique que la $\forall$-règle peut être appliquée sans violer (*).

• La $\forall_+$-règle : Si $\forall S.C \in L(x)$, alors $\forall S.C \in L(\pi(x))$, et $(\pi(x))^{\mathcal{I}} \in (\forall S.C)^{\mathcal{I}}$. De plus, s'il y a un R avec $R \sqsubseteq S$ et $\mathsf{Trans}(R)$, et y est un R-voisin de x, alors $\langle(\pi(x))^{\mathcal{I}}, (\pi(y))^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$ en raison de (*). Supposons que $(\pi(y))^{\mathcal{I}} \notin (\forall R.C)^{\mathcal{I}}$. À cause de la définition de $(\forall R.C)^{\mathcal{I}}$, il existe $\langle(\pi(y))^{\mathcal{I}}, t^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$ telle que $t^{\mathcal{I}} \notin C^{\mathcal{I}}$. Puisque $R \sqsubseteq S$ et $\mathsf{Trans}(R)$, nous avons $\langle(\pi(x))^{\mathcal{I}}, (\pi(y))^{\mathcal{I}}\rangle, \langle(\pi(y))^{\mathcal{I}}, t^{\mathcal{I}}\rangle, \langle(\pi(x))^{\mathcal{I}}, t^{\mathcal{I}}\rangle \in R^{\mathcal{I}} \subseteq$

$S^{\mathcal{I}}$. Donc, $\langle(\pi(x))^{\mathcal{I}}, t^{\mathcal{I}}\rangle \in S^{\mathcal{I}}$ et $t^{\mathcal{I}} \notin C^{\mathcal{I}}$ contredisent $(\pi(x))^{\mathcal{I}} \in (\forall S.C)^{\mathcal{I}}$. Alors, $(\pi(y))^{\mathcal{I}} \in (\forall R.C)^{\mathcal{I}}$ et donc la $\forall_+$-règle peut être appliquée sans violer (*).

• La $\geq$-règle : Si $(\geq nSC) \in L(x)$, alors $(\geq nSC) \in \mathcal{L}(\pi(x))$ et $(\pi(x))^{\mathcal{I}} \in (\geq nS.C)^{\mathcal{I}}$. À cause de la définition de $(\geq nS.C)^{\mathcal{I}}$, il y a des individus $t_1, \ldots, t_n \in \Delta$ tels que $\langle(\pi(x))^{\mathcal{I}}, t_i^{\mathcal{I}}\rangle \in S^{\mathcal{I}}$, $t_i^{\mathcal{I}} \in C^{\mathcal{I}}$ et $t_i^{\mathcal{I}} \neq t_j^{\mathcal{I}}$ pour $1 \leq i < j \leq n$. La $\geq$-règle génère n de nouveaux nœuds $y_1, \ldots, y_n$ avec $L(\langle x, y_i\rangle) = \{S\}$ et $L(y_i) = X$ pour un $X \in \mathsf{Flat}(C)$ tel que $t_i^{\mathcal{I}} \in X^{\mathcal{I}} \subseteq C^{\mathcal{I}}$. En mettant $\pi' := \pi[y_1 \to t_1, \ldots, y_n \to t_n]$, on obtient une fonction π' qui satisfait (*) pour la forêt modifiée.

• La $\leq$-règle : Si $(\leq nSC) \in L(x)$, alors $(\leq nSC) \in L(\pi(x))$ et $(\pi(x))^{\mathcal{I}} \in (\leq nS.C)^{\mathcal{I}}$. Si la $\leq$-règle est applicable, alors il y a au moins $n+1$ S-voisins $y_0, \ldots, y_n$ de x tels que $X_i \subseteq L(y_i)$ pour un $X_i \in \mathsf{Flat}(C)$. En plus, il existe deux nœuds $y, z \in \{y_0, \ldots, y_n\}$ tels que $(\pi(y))^{\mathcal{I}} = (\pi(z))^{\mathcal{I}}$ (car sinon $(\pi(x))^{\mathcal{I}} \in (\geq nS.C)^{\mathcal{I}}$ est satisfait). $(\pi(y))^{\mathcal{I}} = (\pi(z))^{\mathcal{I}}$ implique que $y \neq z$ ne peut pas être satisfait en raison de (*) et y, z peut être choisis tels que y n'est pas un nœud racine ni un ancêtre de z.Alors la $\leq$-règle peut être appliquée sans violer (*).

• La $\leq_r$-règle : Si $(\leq nSC) \in L(x)$, alors $(\leq nSC) \in L(\pi(x))$ et $(\pi(x))^{\mathcal{I}} \in (\leq nS.C)^{\mathcal{I}}$. Si la $\leq_r$-règle est applicable, alors il y a au moins $n+1$ S-voisins $y_0, \ldots, y_n$ de x tels que $X_i \subseteq L(y_i)$ pour un $X_i \in \mathsf{Flat}(C)$. Donc, il existe deux nœuds racines $y, z \in \{y_0, \ldots, y_n\}$ tels que $(\pi(y))^{\mathcal{I}} = (\pi(z))^{\mathcal{I}}$ (car sinon $(\pi(x))^{\mathcal{I}} \in (\geq nS.C)^{\mathcal{I}}$ est satisfait). $(\pi(y))^{\mathcal{I}} = (\pi(z))^{\mathcal{I}}$ implique que $y \neq z$ ne peut pas être satisfait en raison de (*). Alors la $\leq_r$-règle peut être appliquée sans violer (*).

• La sat-règle : Si la sat-règle n'est jamais appliquée à x alors cette règle peut être appliquée sans violer (*) en choisissant $S = \bar{S} = L(\pi(x))$.

Nous montrons que l'affirmation donne la complétude du nouvel algorithme de tableau. Nous pouvons initialiser une forêt de complétion consistant en des nœuds racines $\widehat{x}_i$ avec $L(\widehat{x}_i) := \{\{a_i\} \cup X_k \in X \mid C(a_i) \in \mathcal{A}, X \in \mathsf{Flat}(C)\}$, $L(\langle\widehat{x}_i, \widehat{x}_j\rangle) := \{R \mid R(a_i, a_j) \in \mathcal{A}\}$, $\widehat{x}_i \neq \widehat{x}_j$ ssi $a_i \neq a_j \in \mathcal{A}$, et la $\doteq$-relation est initialisée à vide. Nous prenons une fonction π satisfaisant (*) en mettant $\pi(\widehat{x}_i) = a_i^{\mathcal{I}}$ pour des individus a_i dans $\mathcal{A}$ et $a_i^{\mathcal{I}} \in \Delta$. Chaque fois qu'une règle est applicable à $\mathcal{F}$, elle peut être appliquée d'une manière qui maintient (*), et, selon la Lemme 4, nous avons qu'une

séquence d'applications de règle doit se terminer. Car (*) est satisfaite, Toute forêt générée par ces applications de règles doit être non-clash. Ceci peut être vu comme suit avec deux possibilités pour un clash :

— $\mathcal{F}$ ne peut pas contenir un nœud x tel que $\{C, \neg C\} \in L(x)$ car $L(x) \subseteq L(\pi(x))$ et donc $(\pi(x))^{\mathcal{I}} \in C^{\mathcal{I}} \cap (\neg C)^{\mathcal{I}}$ devrait être violé selon la définition de $C^{\mathcal{I}}$ et $(\neg C)^{\mathcal{I}}$.

— $\mathcal{F}$ ne peut pas contenir un nœud x avec $(\leq nSC) \in L(x)$ et $n+1$ S-voisins $y_0, \ldots, y_n$ de x avec $X \subseteq L(y_i)$ et $y_i \not\equiv y_j$ pour un $X \in \mathsf{Flat}(C)$ et $0 \leq i < j \leq n$, car $(\leq nSC) \in L(\pi(x))$, et, puisque $y_i \not\equiv y_j$ implique $(\pi(y_i))^{\mathcal{I}} \neq (\pi(y_j))^{\mathcal{I}}$, $(\pi(x))^{\mathcal{I}} \in (\leq nSC)^{\mathcal{I}}$ est satisfaite qui contredit la définition de $(\leq nSC)^{\mathcal{I}}$. □

□

Lemme 5. *La fonction $d(\mathcal{F}, \mathcal{F}')$ dans la définition 12 satisfait les propriétés d'identité, de symétrie et d'inégalité triangulaire.*

Démonstration. Nous devons prouver les propriétés suivantes :

• **symétrique**. En raison de la commutativité de l'opérateur $\triangle$, nous avons $L(\langle \widehat{x}, \widehat{y} \rangle) \triangle L'(\langle \varphi(\widehat{x}), \varphi(\widehat{y}) \rangle) = L'(\langle \varphi(\widehat{x}), \varphi(\widehat{y}) \rangle) \triangle L(\langle \widehat{x}, \widehat{y} \rangle)$. De plus, $d(T\langle \widehat{x}_i \rangle, T\langle \varphi(\widehat{x}_i) \rangle) = d(T\langle \varphi(\widehat{x}_i) \rangle, T\langle \widehat{x}_i \rangle)$ selon la définition de $d(T\langle \widehat{x}_i \rangle, T\langle \varphi(\widehat{x}_i) \rangle)$ utilise aussi l'opérateur $\triangle$ sur $L(x, y) \triangle L(\pi(x), \pi(y))$ pour $\pi \in \Pi(T\langle \widehat{x}_i \rangle, T\langle \varphi(\widehat{x}_i) \rangle)$. Alors, $d(\mathcal{F}, \mathcal{F}') = d(\mathcal{F}', \mathcal{F})$.

• **identité**. Selon le fait que $S \triangle S' = \emptyset$ ssi $S = S'$ pour tous les ensembles S, S', nous avons que $|L(\langle \widehat{x}, \widehat{y} \rangle) \triangle L'(\langle \varphi(\widehat{x}), \varphi(\widehat{y}) \rangle)| = \max_{\langle x,y \rangle \in \mathbf{E}} (|L(\langle x, y \rangle) \triangle L'(\langle \varphi(x), \varphi(y) \rangle)|) = 0$ ssi $L(\langle \widehat{x}, \widehat{y} \rangle) = L'(\langle \varphi(\widehat{x}), \varphi(\widehat{y}) \rangle)$.

Comme $d(T\langle \widehat{x}_i \rangle, T\langle \varphi(\widehat{x}_i) \rangle)$ est également définie en utilisant les opérateurs $\triangle$ et max, nous avons aussi $d(T\langle \widehat{x}_i \rangle, T\langle \varphi(\widehat{x}_i) \rangle) = 0$ ssi $T\langle \widehat{x}_i \rangle = T\langle \varphi(\widehat{x}_i) \rangle$. Alors, $d(\mathcal{F}, \mathcal{F}') = 0$ ssi $\mathcal{F} = \mathcal{F}'$.

• **triangle inégalité**. D'abord, nous devons démontrer que $d(T, T'') \leq d(T, T') + d(T', T'')$ où $T = T\langle \widehat{x} \rangle = \langle V, L, E \rangle$, $T' = T\langle \widehat{x}' \rangle = \langle V', L', E' \rangle$ et $T'' = T\langle \widehat{x}'' \rangle = \langle V'', L'', E'' \rangle$. Par la définition de distance, nous avons :

$$d(T, T'') = \min_{\pi \in \Pi(T, T'')} \{ \max_{\langle x,y \rangle \in E} (|L(x) \triangle L''(\pi(x))| +$$

$|L(\langle x,y\rangle) \vartriangle L''(\langle \pi(x),\pi(y)\rangle)| +$
$|L(y) \vartriangle L''(\pi(y))|)\}$

Cela implique que :

1. Pour chaque $\pi \in \Pi(T,T'')$, nous pouvons choisir $\langle x_0,y_0\rangle$ de E telle que :

   $|L(x_0) \vartriangle L''(\pi(x_0))|$
   $+|L(\langle x_0,y_0\rangle) \vartriangle L''(\langle \pi(x_0),\pi(y_0)\rangle)|$
   $+|L(y_0) \vartriangle L''(\pi(y_0))| = \max\limits_{\langle x,y\rangle\in E}(|L(x) \vartriangle L''(\pi(x))|$
   $+|L(\langle x,y\rangle) \vartriangle L''(\langle \pi(x),\pi(y)\rangle)|$
   $+|L(y) \vartriangle L''(\pi(y))|) = max_0(\pi)$

   Donc, $d(T,T'') = \min\limits_{\pi\in\Pi(T,T'')}(max_0(\pi))$.

   En plus, nous pouvons choisir π_1 de $\Pi(T,T'')$ telle que $d(T,T'') = max_0$ avec $max_0 = max_0(\pi_1)$.

2. De même, nous pouvons choisir $\langle x_1,y_1\rangle$ de E telle que $d(T,T') = \min\limits_{\pi\in\Pi(T,T')}(max_1(\pi))$
   avec $max_1(\pi) = \max\limits_{\langle x,y\rangle\in E}(|L(x) \vartriangle L'(\pi(x))|$
   $+|L(\langle x,y\rangle) \vartriangle L'(\langle \pi(x),\pi(y)\rangle)|$
   $+|L(y) \vartriangle L'(\pi(y))|) = max_1(\pi)$

   et choisir π_2 de $\Pi(T,T')$ telle que $d(T,T') = max_1$ avec $max_1 = max_1(\pi_2)$.

3. De même, nous pouvons choisir $\langle x_2,y_2\rangle$ de E' telle que $d(T',T'') = \min\limits_{\pi\in\Pi(T',T'')}(max_2(\pi))$
   avec $max_2(\pi) = \max\limits_{\langle x,y\rangle\in E'}(|L'(x) \vartriangle L''(\pi(x))|$
   $+|L'(\langle x,y\rangle) \vartriangle L''(\langle \pi(x),\pi(y)\rangle)|$
   $+|L'(y) \vartriangle L''(\pi(y))|) = max_2(\pi)$

   et choisir π_3 de $\Pi(T',T'')$ telle que $d(T',T'') = max_2$ avec $max_2 = max_2(\pi_3)$.

Selon le fait que $S \vartriangle S'' \subseteq (S \vartriangle S') \cup (S' \vartriangle S'')$, pour toute $\pi \in \Pi(T,T'')$ et $\langle x,y\rangle \in E$ nous avons

$|L(x) \vartriangle L''(\pi(x))| + |L(\langle x,y\rangle) \vartriangle L''(\langle \pi(x),\pi(y)\rangle)|$
$+|L(y) \vartriangle L''(\pi(y))| \leq$

$|L(x) \vartriangle L'(\pi_2(x))| + |L'(\pi_2(x)) \vartriangle L''(\pi(x))|$

$+|L(\langle x,y\rangle) \vartriangle L'(\langle \pi_2(x),\pi_2(y)\rangle)|$
$+|L'(\langle \pi_2(x),\pi_2(y)\rangle) \vartriangle L''(\langle \pi(x),\pi(y)\rangle)|$
$+|L(y) \vartriangle L'(\pi_2(y))|$
$+|L'(\pi_2(y)) \vartriangle L''(\pi(y))| \leq$

$|L(x) \vartriangle L'(\pi_2(x))| + |L'(\pi_2(x)) \vartriangle L''(\pi_3(\pi_2(x)))|$
$+|L''(\pi_3(\pi_2(x))) \vartriangle L''(\pi(x))|$
$+|L(\langle x,y\rangle) \vartriangle L'(\langle \pi_2(x),\pi_2(y)\rangle)|$
$+|L'(\langle \pi_2(x),\pi_2(y)\rangle) \vartriangle L''(\langle \pi_3(\pi_2(x)),\pi_3(\pi_2(y))\rangle)|$
$+|L''(\langle \pi_3(\pi_2(x)),\pi_3(\pi_2(y))\rangle) \vartriangle L''(\langle \pi(x),\pi(y)\rangle)|$
$+|L(y) \vartriangle L'(\pi_2(y))|$
$+|L'(\pi_2(y)) \vartriangle L''(\pi_3(\pi_2(y)))|$
$+|L''(\pi_3(\pi_2(y))) \vartriangle L''(\pi(y))|$ $(*)$.

Puisque $max_1 = max_1(\pi_2) = \max\limits_{\langle x,y\rangle \in E}(|L(x) \vartriangle L'(\pi_2(x))|$
$+|L(\langle x,y\rangle) \vartriangle L'(\langle \pi_2(x),\pi_2(y)\rangle)|$
$+|L(y) \vartriangle L'(\pi_2(y))|$

et $max_2 = max_2(\pi_3) = \max\limits_{\langle x,y\rangle \in E'}(|L'(x) \vartriangle L''(\pi_3(x))|$
$+|L'(\langle x,y\rangle) \vartriangle L''(\langle \pi_3(x),\pi_3(y)\rangle)|$
$+|L'(y) \vartriangle L''(\pi_3(y))|$,

nous avons $|L(x) \vartriangle L'(\pi_2(x))|$
$+|L(\langle x,y\rangle) \vartriangle L'(\langle \pi_2(x),\pi_2(y)\rangle)|$
$+|L(y) \vartriangle L'(\pi_2(y))| \leq max_1$,

et $|L'(\pi_2(x)) \vartriangle L''(\pi_3(\pi_2(x)))|$
$+|L'(\langle \pi_2(x),\pi_2(y)\rangle) \vartriangle L''(\langle \pi_3(\pi_2(x)),\pi_3(\pi_2(y))\rangle)|$
$+|L'(\pi_2(y)) \vartriangle L''(\pi_3(\pi_2(y)))| \leq max_2$ pour toute $\langle x,y\rangle \in E$.

Par conséquent, $(*)$ est écrite comme suit :

$|L(x) \vartriangle L''(\pi(x))| + |L(\langle x,y\rangle) \vartriangle L''(\langle \pi(x),\pi(y)\rangle)| + |L(y) \vartriangle L''(\pi(y))| \leq max_1 + max_2 + |L''(\pi_3(\pi_2(x))) \vartriangle L''(\pi(x))| + |L''(\langle \pi_3(\pi_2(x)),\pi_3(\pi_2(y))\rangle) \vartriangle L''(\langle \pi(x),\pi(y)\rangle)| + |L''(\pi_3(\pi_2(y))) \vartriangle L''(\pi(y))|$ pour toute $\pi \in \Pi(T,T'')$ et toute $\langle x,y\rangle \in E$ $(**)$.

Comme π_2 et π_3 sont bijectives, il existe toujours $\hat{\pi}_1 \in \Pi(T, T'')$ telle que $\hat{\pi}_1 = \pi_3 \circ \pi_2$. Cela implique que π dans $(**)$ peut être remplacée par $\hat{\pi}_1$ pour obtenir $|L''(\pi_3(\pi_2(x))) \vartriangle L''(\pi(x))| + |L''(\langle \pi_3(\pi_2(x)), \pi_3(\pi_2(y))\rangle) \vartriangle L''(\langle \pi(x), \pi(y)\rangle)| + |L''(\pi_3(\pi_2(y))) \vartriangle L''(\pi(y))| = 0$ et donc $|L(x) \vartriangle L''(\hat{\pi}_1(x))| + |L(\langle x, y\rangle) \vartriangle L''(\langle \hat{\pi}_1(x), \hat{\pi}_1(y)\rangle)| + |L(y) \vartriangle L''(\hat{\pi}_1(y))| \leq max_1 + max_2$ (i) pour toute $\langle x, y\rangle \in E$.

En plus, il existe $\langle \hat{x}_1, \hat{y}_1\rangle \in E$ pour $\hat{\pi}_1$ telle que :

$|L(\hat{x}_1) \vartriangle L''(\hat{\pi}_1(\hat{x}_1))| + |L(\langle \hat{x}_1, \hat{y}_1\rangle) \vartriangle L''(\langle \hat{\pi}_1(\hat{x}_1), \hat{\pi}_1(\hat{y}_1)\rangle)| + |L(\hat{y}_1) \vartriangle L''(\hat{\pi}_1(\hat{y}_1))| =$
$\max\limits_{\langle x,y\rangle \in E}(|L(x) \vartriangle L''(\hat{\pi}_1(x))| + |L(\langle x, y\rangle) \vartriangle L''(\langle \hat{\pi}_1(x), \hat{\pi}_1(y)\rangle)|$
$+|L(y) \vartriangle L''(\hat{\pi}_1(y))|)$ (ii),

Par le point 1., nous avons :

$d(T, T'') = \min\limits_{\pi \in \Pi(T,T'')}\{\max\limits_{\langle x,y\rangle \in E}(|L(x) \vartriangle L''(\pi(x))| +$
$|L(\langle x, y\rangle) \vartriangle L''(\langle \pi(x), \pi(y)\rangle)| +$
$|L(y) \vartriangle L''(\pi(y))|)\}$

Cela implique que
$d(T, T'') \leq \max\limits_{\langle x,y\rangle \in E}(|L(x) \vartriangle L''(\hat{\pi}_1(x))|$
$+|L(\langle x, y\rangle) \vartriangle L''(\langle \hat{\pi}_1(x), \hat{\pi}_1(y)\rangle)|$
$+|L(y) \vartriangle L''(\hat{\pi}_1(y))|)$ car $\hat{\pi}_1 \in \Pi(T, T'')$.
Cela signifie que $d(T, T'') \leq |L(\hat{x}_1) \vartriangle L''(\hat{\pi}_1(\hat{x}_1))| + |L(\langle \hat{x}_1, \hat{y}_1\rangle) \vartriangle L''(\langle \hat{\pi}_1(\hat{x}_1), \hat{\pi}_1(\hat{y}_1)\rangle)| + |L(\hat{y}_1) \vartriangle L''(\hat{\pi}_1(\hat{y}_1))|$ selon (ii). En conséquence de (i), $d(T, T'') \leq max_1 + max_2$. Cela signifie que $d(T, T'') \leq d(T, T') + d(T', T'')$.

Nous remplaçons T, T' et T'' dans la dernière inégalité avec $T\langle \widehat{x}_i\rangle$, $T\langle \varphi_2(\widehat{x}_i)\rangle$ et $T\langle \varphi_1(\widehat{x}_i)\rangle$ respectivement où φ_1 est une bijection de $\mathbf{V}$ à $\mathbf{V}''$, φ_2 est une bijection de $\mathbf{V}$ à $\mathbf{V}'$ (selon la Remarque 1) avec $\mathbf{V}$, $\mathbf{V}'$, $\mathbf{V}''$ les ensembles de nœuds racines de $T\langle \widehat{x}_i\rangle$, $T\langle \varphi_2(\widehat{x}_i)\rangle$ et $T\langle \varphi_1(\widehat{x}_i)\rangle$. Alors, nous obtenons

$d(T\langle \widehat{x}_i\rangle, T\langle \varphi_1(\widehat{x}_i)\rangle) \leq d(T\langle \widehat{x}_i\rangle, T\langle \varphi_2(\widehat{x}_i)\rangle)$
$+d(T\langle \varphi_2(\widehat{x}_i)\rangle, T\langle \varphi_1(\widehat{x}_i)\rangle)$

pour toute $T\langle \widehat{x}_i\rangle \in \mathcal{F}$, $T\langle \varphi_1(\widehat{x}_i)\rangle \in \mathcal{F}''$ et $T\langle \varphi_2(\widehat{x}_i)\rangle \in \mathcal{F}'$ $(1 \leq i \leq n)$. Cela implique que

$$\sum_{i=1}^{n} d(T\langle \widehat{x}_i\rangle, T\langle \varphi_1(\widehat{x}_i)\rangle) \leq \sum_{i=1}^{n} d(T\langle \widehat{x}_i\rangle, T\langle \varphi_2(\widehat{x}_i)\rangle)$$
$$+\sum_{i=1}^{n} d(T\langle \varphi_2(\widehat{x}_i)\rangle, T\langle \varphi_1(\widehat{x}_i)\rangle) \ (**)$$

Ensuite, nous devons démontrer que

$$\max_{\langle \widehat{x},\widehat{y}\rangle \in \mathbf{E}} (|L(\langle \widehat{x},\widehat{y}\rangle) \vartriangle L''(\langle \varphi_1(\widehat{x}), \varphi_1(\widehat{y})\rangle)|) \ \leq$$
$$\max_{\langle \widehat{x},\widehat{y}\rangle \in \mathbf{E}} (|L(\langle \widehat{x},\widehat{y}\rangle) \vartriangle L'(\langle \varphi_2(\widehat{x}), \varphi_2(\widehat{y})\rangle)|) \ +$$
$$\max_{\langle \widehat{x},\widehat{y}\rangle \in \mathbf{E}'} (|L'(\langle \widehat{x},\widehat{y}\rangle)|) \vartriangle L''(\langle \varphi_3(\widehat{x}), \varphi_3(\widehat{y})\rangle)|)$$

où φ_1 est une bijection de $\mathbf{V}$ à $\mathbf{V}''$, φ_2 est une bijection de $\mathbf{V}$ à $\mathbf{V}'$ et φ_3 est une bijection de $\mathbf{V}'$ à $\mathbf{V}''$ avec $\mathbf{V}$, $\mathbf{V}'$, $\mathbf{V}''$ les ensembles de nœuds racines de $\mathcal{F}$, $\mathcal{F}'$ et $\mathcal{F}''$; $\mathbf{E}$ et $\mathbf{E}'$ sont les ensembles d'arêtes entre des nœuds racines dans $\mathcal{F}$ et $\mathcal{F}'$.

i. Nous pouvons choisir $\langle \widehat{x}_1, \widehat{y}_1\rangle$ de $\mathbf{E}$ telle que

$$|L(\langle \widehat{x}_1, \widehat{y}_1\rangle) \vartriangle L''(\langle \varphi_1(\widehat{x}_1), \varphi_1(\widehat{y}_1)\rangle)| =$$
$$\max_{\langle \widehat{x},\widehat{y}\rangle \in \mathbf{E}} (|L(\langle \widehat{x},\widehat{y}\rangle) \vartriangle L''(\langle \varphi_1(\widehat{x}), \varphi_1(\widehat{y})\rangle)|) = max_1.$$

ii. Nous mettons

$$max_2 = \max_{\langle \widehat{x},\widehat{y}\rangle \in \mathbf{E}} (|L(\langle \widehat{x},\widehat{y}\rangle) \vartriangle L'(\langle \varphi_2(\widehat{x}), \varphi_2(\widehat{y})\rangle)|),$$
et
$$max_3 = \max_{\langle \widehat{x},\widehat{y}\rangle \in \mathbf{E}'} (|L'(\langle \widehat{x},\widehat{y}\rangle) \vartriangle L''(\langle \varphi_3(\widehat{x}), \varphi_3(\widehat{y})\rangle)|).$$

iii. Selon la Remarque 1, pour chaque individu a il y a un seul arbre $T\langle \widehat{x}_i\rangle$ de $\mathcal{F}$, un seul arbre $T\langle \widehat{x}'_j\rangle$ de $\mathcal{F}'$ et un seul arbre $T\langle \widehat{x}''_k\rangle$ de $\mathcal{F}''$ tels que $a \in L(\widehat{x}_i) \cap L'(\widehat{x}'_j) \cap L''(\widehat{x}''_k)$, alors nous avons que pour chaque $\widehat{x}_i \in \mathbf{V}$, $\varphi_1(\widehat{x}_i) = \varphi_3(\varphi_2(\widehat{x}_i))$.

Puisque $|S \vartriangle S''| \leq |S \vartriangle S'| + |S' \vartriangle S''|$, nous avons que

$$max_1 = |L(\langle \widehat{x}_1, \widehat{y}_1\rangle) \vartriangle L''(\langle \varphi_1(\widehat{x}_1), \varphi_1(\widehat{y}_1)\rangle)| \ \leq$$
$$|L(\langle \widehat{x}_1, \widehat{y}_1\rangle) \vartriangle L'(\langle \varphi_2(\widehat{x}_1), \varphi_2(\widehat{y}_1)\rangle)| +$$
$$|L'(\langle \varphi_2(\widehat{x}_1), \varphi_2(\widehat{y}_1)\rangle) \vartriangle L''(\langle \varphi_1(\widehat{x}_1), \varphi_1(\widehat{y}_1)\rangle)|.$$

À partir des points ii. et iii., nous obtenons

$$|L(\langle \widehat{x}_1, \widehat{y}_1\rangle) \vartriangle L'(\langle \varphi_2(\widehat{x}_1), \varphi_2(\widehat{y}_1)\rangle)| \leq max_2 \text{ et } |L'(\langle \varphi_2(\widehat{x}_1), \varphi_2(\widehat{y}_1)\rangle) \vartriangle L''(\langle \varphi_1(\widehat{x}_1), \varphi_1(\widehat{y}_1)\rangle)| \ =$$

$|L'(\langle\varphi_2(\widehat{x}_1),\varphi_2(\widehat{y}_1)\rangle) \vartriangle L''(\langle\varphi_3(\varphi_2(\widehat{x}_1)),\varphi_3(\varphi_2(\widehat{y}_1))\rangle)| \leq max_3$.

Alors, $max_1 \leq max_2 + max_3$ (***).

Selon (**) et (***), $d(\mathcal{F},\mathcal{F}'') \leq d(\mathcal{F},\mathcal{F}') + d(\mathcal{F}',\mathcal{F}'')$. □

Théorème 4. *L'opération de révision* $\mathsf{FM}(\mathcal{O},\mathcal{O}')$ *décrite dans la définition 13 satisfait les postulats* ***(P1)-(P6)****.*

Démonstration.

— **(P1)** Soit $\mathcal{F} \in \mathsf{FM}(\mathcal{O},\mathcal{O}')$. Selon la Définition 13 disant que $\mathsf{FM}(\mathcal{O},\mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O}))$, nous avons $\mathcal{F} \in \mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O}))$. Cela signifie que $\mathcal{I}(\mathcal{F})$ est un modèle de $\mathcal{O}'$. Donc, $\mathcal{I}(\mathcal{F}) \models \alpha$ pour chaque axiome $\alpha \in \mathcal{O}'$. Alors, $\mathcal{I}(\mathcal{F}) \models \alpha$ pour chaque modèle de forêt $\mathcal{F} \in \mathsf{FM}(\mathcal{O},\mathcal{O}')$ et chaque axiome $\alpha \in \mathcal{O}'$.

— **(P2)** Soit $\mathcal{F} \in \mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O}))$. Alors $\mathcal{F} \in \mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O}))$ et $\mathcal{F} \in \mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}'))$. Par la Définition 12, nous avons $d(\mathcal{F},\mathcal{F}) = 0$. Nous devons démontrer que $\mathcal{F} \in \mathsf{FM}(\mathcal{O},\mathcal{O}')$. Puisque $d(\mathcal{F},\mathcal{F}) = 0$, pour tout modèle de forêt $\mathcal{F}_i \in \mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}'))$ et $\mathcal{F}_j \in \mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O}))$, nous avons $d(\mathcal{F},\mathcal{F}) \leq d(\mathcal{F}_i,\mathcal{F}_j)$. Cela implique que $\mathcal{F} \in \mathsf{FM}(\mathcal{O},\mathcal{O}')$ par la Définition 13. Alors, $\mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O})) \subseteq \mathsf{FM}(\mathcal{O},\mathcal{O}')$ (*).

Nous montrons que $\mathsf{FM}(\mathcal{O},\mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O}))$. Soit $\mathcal{F}_0 \in \mathsf{FM}(\mathcal{O},\mathcal{O}')$. Par la Définition 13, nous avons $\mathcal{F}_0 \in \mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O}))$. Supposons que $\mathcal{F}_0 \notin \mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}'))$ (l'hypothèse d'absurdité). Cela implique que $\mathcal{F}_0$ est différent de chaque modèle de forêt $\mathcal{F}' \in \mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}'))$. Par la Définition 12, $d(\mathcal{F}_0,\mathcal{F}') \neq 0$ pour chaque $\mathcal{F}' \in \mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}'))$. En plus, nous avons $\mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O})) \cap \mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}')) \neq \emptyset$ par l' hypothèse de **(P2)**. Donc, il existe un modèle de forêt $\mathcal{F}_1 \in \mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O})) \cap \mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}'))$ avec $d(\mathcal{F}_1,\mathcal{F}_1) = 0$, et alors $d(\mathcal{F}_1,\mathcal{F}_1) < d(\mathcal{F}_0,\mathcal{F}')$. Par la Définition 13, nous avons $\mathcal{F}_0 \notin \mathsf{FM}(\mathcal{O},\mathcal{O}')$, qui contredit $\mathcal{F}_0 \in \mathsf{FM}(\mathcal{O},\mathcal{O}')$. Par absurdité, nous obtenons $\mathcal{F}_0 \in \mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}'))$, et donc, $\mathcal{F}_0 \in \mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O}))$. Alors, $\mathsf{FM}(\mathcal{O},\mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O}))$ (**). Selon (*) & (**), nous obtenons $\mathsf{FM}(\mathcal{O},\mathcal{O}') = \mathsf{FM}(\mathcal{O},\mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O}))$.

— **(P3)** Supposons que $\mathsf{FM}(\mathcal{O},\mathcal{O}') = \emptyset$ (l'hypothèse d'absurdité). Puisque $\mathcal{O}'$ est cohérente, $\mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O})) \neq \emptyset$. Avec $\mathsf{FM}(\mathcal{O},\mathcal{O}') = \emptyset$, pour chaque $\mathcal{F}'_0 \in \mathsf{FM}(\mathcal{O}',\mathsf{sub}(\mathcal{O}))$

et $\mathcal{F}_0 \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$, il existe toujours une couple de modèles de forêt $\mathcal{F}'_i \in \mathsf{FM}(\mathcal{O}', \mathsf{sub})$ et $\mathcal{F}_i \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ telle que $d(\mathcal{F}'_i, \mathcal{F}_i) < d(\mathcal{F}'_0, \mathcal{F}_0)$. et pour chaque couple de $\mathcal{F}'_i$ et $\mathcal{F}_i$, nous pouvons déterminer une autre couple $\mathcal{F}'_j \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ et $\mathcal{F}_j \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ avec $i \neq j$ telle que $d(\mathcal{F}'_j, \mathcal{F}_j) < d(\mathcal{F}'_i, \mathcal{F}_i)$.

Cependant, le nombre de forêts dans $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ et $\mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ est fini. Cela implique qu'il existe une couple $\mathcal{F}'_j \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ et $\mathcal{F}_j \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ telle qu'il n'existe pas une autre couple $\mathcal{F}'_n \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ et $\mathcal{F}_n \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ telle que $d(\mathcal{F}'_n, \mathcal{F}_n) < d(\mathcal{F}'_j, \mathcal{F}_j)$. Cela implique que $d(\mathcal{F}'_j, \mathcal{F}_j) \leq d(\mathcal{F}'_n, \mathcal{F}_n)$ pour toute $\mathcal{F}'_n \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ et $\mathcal{F}_n \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$. Selon la Définition 13, nous obtenons $\mathcal{F}'_j \in \mathsf{FM}(\mathcal{O}, \mathcal{O}')$. Alors, $\mathsf{FM}(\mathcal{O}, \mathcal{O}') \neq \emptyset$, qui contredit l'hypothèse d'absurdité.

— **(P4)** Soit $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}_1, \mathcal{O}'_1)$. Par la Définition 13, $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}'_1, \mathsf{sub}(\mathcal{O}_1))$ et il existe $\mathcal{F} \in \mathsf{FM}(\mathcal{O}_1, \mathsf{sub}(\mathcal{O}'_1))$ telle que $d(\mathcal{F}', \mathcal{F}) \leq d(\mathcal{F}_i, \mathcal{F}_j)$ pour chaque $\mathcal{F}_i \in \mathsf{FM}(\mathcal{O}'_1, \mathsf{sub}(\mathcal{O}_1))$ et $\mathcal{F}_j \in \mathsf{FM}(\mathcal{O}_1, \mathsf{sub}(\mathcal{O}'_1))$. Par l'hypothèse de **(P4)**, nous pouvons remplacer $\mathsf{FM}(\mathcal{O}'_1, \mathsf{sub}(\mathcal{O}_1))$ avec $\mathsf{FM}(\mathcal{O}'_2, \mathsf{sub}(\mathcal{O}_2))$ et $\mathsf{FM}(\mathcal{O}_1, \mathsf{sub}(\mathcal{O}'_1))$ avec $\mathsf{FM}(\mathcal{O}_2, \mathsf{sub}(\mathcal{O}'_2))$. Donc, nous obtenons $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}'_2, \mathsf{sub}(\mathcal{O}_2))$ et il existe $\mathcal{F} \in \mathsf{FM}(\mathcal{O}_2, \mathsf{sub}(\mathcal{O}'_2))$ telle que $d(\mathcal{F}', \mathcal{F}) \leq d(\mathcal{F}_i, \mathcal{F}_j)$ pour chaque $\mathcal{F}_i \in \mathsf{FM}(\mathcal{O}'_2, \mathsf{sub}(\mathcal{O}_2))$ et $\mathcal{F}_j \in \mathsf{FM}(\mathcal{O}_2, \mathsf{sub}(\mathcal{O}'_2))$. Cela implique $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}_2, \mathcal{O}'_2)$. Alors, $\mathsf{FM}(\mathcal{O}_1, \mathcal{O}'_1) \subseteq \mathsf{FM}(\mathcal{O}_2, \mathcal{O}'_2)$.

Soit $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}_2, \mathcal{O}'_2)$. Par la Définition 13, $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}'_2, \mathsf{sub}(\mathcal{O}_2))$ et il existe $\mathcal{F} \in \mathsf{FM}(\mathcal{O}_2, \mathsf{sub}(\mathcal{O}'_2))$ telle que $d(\mathcal{F}', \mathcal{F}) \leq d(\mathcal{F}_i, \mathcal{F}_j)$ pour chaque $\mathcal{F}_i \in \mathsf{FM}(\mathcal{O}'_2, \mathsf{sub}(\mathcal{O}_2))$ et $\mathcal{F}_j \in \mathsf{FM}(\mathcal{O}_2, \mathsf{sub}(\mathcal{O}'_2))$. Par l'hypothèse de **(P4)**, nous pouvons remplacer $\mathsf{FM}(\mathcal{O}'_2, \mathsf{sub}(\mathcal{O}_2))$ avec $\mathsf{FM}(\mathcal{O}'_1, \mathsf{sub}(\mathcal{O}_1))$ et $\mathsf{FM}(\mathcal{O}_2, \mathsf{sub}(\mathcal{O}'_2))$ avec $\mathsf{FM}(\mathcal{O}_1, \mathsf{sub}(\mathcal{O}'_1))$. Donc, nous obtenons $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}'_1, \mathsf{sub}(\mathcal{O}_1))$ et il existe $\mathcal{F} \in \mathsf{FM}(\mathcal{O}_1, \mathsf{sub}(\mathcal{O}'_1))$ telle que $d(\mathcal{F}', \mathcal{F}) \leq d(\mathcal{F}_i, \mathcal{F}_j)$ pour chaque $\mathcal{F}_i \in \mathsf{FM}(\mathcal{O}'_1, \mathsf{sub}(\mathcal{O}_1))$ et $\mathcal{F}_j \in \mathsf{FM}(\mathcal{O}_1, \mathsf{sub}(\mathcal{O}'_1))$. Cela implique $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}_1, \mathcal{O}'_1)$. Donc, $\mathsf{FM}(\mathcal{O}_2, \mathcal{O}'_2) \subseteq \mathsf{FM}(\mathcal{O}_1, \mathcal{O}'_1)$. Alors, $\mathsf{FM}(\mathcal{O}_1, \mathcal{O}'_1) = \mathsf{FM}(\mathcal{O}_2, \mathcal{O}'_2)$.

— **(P5)** Soit $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathcal{O}') \cap \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}'))$.

Par la Définition 13, nous avons $\mathcal{F} \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ car $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathcal{O}')$. Donc, $\mathcal{F} \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \cap \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}'))$ (i).

En appliquant les sat-,$\mathsf{sat}_{\mathcal{R}}$-règles pour construire $\mathcal{F} \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$, nous devons

utiliser l'ensemble $\mathsf{sub}(\mathcal{O}') \cup \mathsf{sub}(\mathcal{O})$ et l'ensemble $\mathbf{R}_{\mathcal{O}}$ respectivement. En appliquant la sat-règle pour construire $\mathcal{F} \in \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}'))$, nous devons utiliser l'ensemble $\mathsf{sub}(\mathcal{O}'') \cup \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$ et l'ensemble $\mathbf{R}_{\mathcal{O}} \cup \mathbf{R}_{\mathcal{O}'}$ respectivement. Donc, $\mathsf{sub}(\mathcal{O}') \cup \mathsf{sub}(\mathcal{O}) = \mathsf{sub}(\mathcal{O}'') \cup \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$ et $\mathbf{R}_{\mathcal{O}} = \mathbf{R}_{\mathcal{O}} \cup \mathbf{R}_{\mathcal{O}'}$. Cela implique que $\mathsf{sub}(\mathcal{O}'') \subseteq \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$ et donc $\mathbf{R}_{\mathcal{O}''} \subseteq \mathbf{R}_{\mathcal{O}'} \subseteq \mathbf{R}_{\mathcal{O}}$, qui nous permet d'obtenir $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}') \cup \mathsf{sub}(\mathcal{O}'')) = \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ (ii).

Selon (i) nous avons $\mathcal{F} \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')) = \mathsf{FM}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}'))$ en raison du Corollaire 2. Donc, $\mathcal{F} \in \mathsf{FM}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub}(\mathcal{O}))$.

Supposons $\mathcal{F} \notin \mathsf{FM}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'')$. Par la Définition 13, il existe $\mathcal{F}_1 \in \mathsf{FM}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub}(\mathcal{O}))$ et $\mathcal{F}_1' \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}') \cup \mathsf{sub}(\mathcal{O}''))$ telles que $d(\mathcal{F}_1, \mathcal{F}_1') < d(\mathcal{F}, \mathcal{F}')$ pour chaque $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}') \cup \mathsf{sub}(\mathcal{O}''))$. De plus, nous avons $\mathcal{F}_1 \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ car $\mathcal{F}_1 \in \mathsf{FM}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub}(\mathcal{O})) = \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \cap \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}))$ en raison du Corollaire 2, nous obtenons aussi $\mathcal{F}', \mathcal{F}_1' \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ en raison de (ii). Tout cela implique $\mathcal{F}_1 \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$, $\mathcal{F}_1' \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ et $d(\mathcal{F}_1, \mathcal{F}_1') < d(\mathcal{F}, \mathcal{F}')$ pour chaque $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$. Par la Définition 13, $\mathcal{F} \notin \mathsf{FM}(\mathcal{O}, \mathcal{O}')$, donc contredisant $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathcal{O}')$. Donc, $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'')$. Alors, $\mathsf{FM}(\mathcal{O}, \mathcal{O}') \cap \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')) \subseteq \mathsf{FM}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'')$.

— **(P6)** Puisque $\mathsf{FM}(\mathcal{O}, \mathcal{O}') \cap \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')) \neq \emptyset$, nous pouvons démontrer $\mathsf{sub}(\mathcal{O}'') \subseteq \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$ et $\mathbf{R}_{\mathcal{O}''} \subseteq \mathbf{R}_{\mathcal{O}'} \subseteq \mathbf{R}_{\mathcal{O}}$ comme dans la preuve de **(P5)**. Comme $\mathsf{sub}(\mathcal{O}'') \subseteq \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$ et $\mathbf{R}_{\mathcal{O}''} \subseteq \mathbf{R}_{\mathcal{O}'}$, nous avons $\mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}') \cup \mathsf{sub}(\mathcal{O}'')) = \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ (i).

Soit $\mathcal{F}_1 \in \mathsf{FM}(\mathcal{O}, \mathcal{O}' \cup \mathcal{O}'')$. Par la Définition 13, nous avons $\mathcal{F}_1 \in \mathsf{FM}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub}(\mathcal{O}))$ et il existe une $\mathcal{F}_1' \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}') \cup \mathsf{sub}(\mathcal{O}''))$ telle que $d(\mathcal{F}_1, \mathcal{F}_1') \leq d(\mathcal{F}_i, \mathcal{F}_j)$ pour toute $\mathcal{F}_i \in \mathsf{FM}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub}(\mathcal{O}))$ et $\mathcal{F}_j \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}') \cup \mathsf{sub}(\mathcal{O}''))$ (*).

Nous devons démontrer $\mathcal{F}_1 \in \mathsf{FM}(\mathcal{O}, \mathcal{O}') \cap \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}'))$.

En raison du Corollaire 2, $\mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \cap \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O})) = \mathsf{FM}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub}(\mathcal{O})) = \mathsf{FM}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')) = \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')) \cap \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}'))$ (ii).

D'abord, puisque $\mathcal{F}_1 \in \mathsf{FM}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub}(\mathcal{O}))$ et (ii), nous avons $\mathcal{F}_1 \in \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup$

$\mathsf{sub}(\mathcal{O}')$).

Ensuite, nous devons montrer que $\mathcal{F}_1 \in \mathsf{FM}(\mathcal{O}, \mathcal{O}')$. Comme $\mathsf{FM}(\mathcal{O}, \mathcal{O}') \cap \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')) \neq \emptyset$, il existe une $\mathcal{F}_0 \in \mathsf{FM}(\mathcal{O}, \mathcal{O}') \cap \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}'))$. Selon $\mathcal{F}_0 \in \mathsf{FM}(\mathcal{O}'', \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}'))$ en raison de (ii), nous obtenons $\mathcal{F}_0 \in \mathsf{FM}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub}(\mathcal{O}))$ (iii).

Selon $\mathcal{F}_0 \in \mathsf{FM}(\mathcal{O}, \mathcal{O}')$, par la Définition 13 il existe $\mathcal{F}'_0 \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ telle que $d(\mathcal{F}_0, \mathcal{F}'_0) \leq d(\mathcal{F}_i, \mathcal{F}_j)$ pour toute $\mathcal{F}_i \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ et $\mathcal{F}_j \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ (iv).

À partir de (iii) et (iv) nous avons $\mathcal{F}_0 \in \mathsf{FM}(\mathcal{O}' \cup \mathcal{O}'', \mathsf{sub}(\mathcal{O}))$ et $\mathcal{F}'_0 \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) = \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}') \cup \mathsf{sub}(\mathcal{O}''))$ en raison de (i). Selon (*), nous obtenons $d(\mathcal{F}_1, \mathcal{F}'_1) \leq d(\mathcal{F}_0, \mathcal{F}'_0)$ (**).

À partir de (*) et (i), nous avons $\mathcal{F}'_1 \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}') \cup \mathsf{sub}(\mathcal{O}'')) = \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$. Supposons que $\mathcal{F}_1 \notin \mathsf{FM}(\mathcal{O}, \mathcal{O}')$. Par la Définition 13 ceci implique qu'il existe $\mathcal{F}_2 \in \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ et $\mathcal{F}'_2 \in \mathsf{FM}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ telles que $d(\mathcal{F}_2, \mathcal{F}'_2) < d(\mathcal{F}_1, \mathcal{F}'_1)$ et $d(\mathcal{F}_0, \mathcal{F}'_0) \leq d(\mathcal{F}_2, \mathcal{F}'_2)$ (en raison de (iv)), qui contredit (**).

□

Théorème 5. *Soient $\mathcal{O}$ et $\mathcal{O}'$ deux ontologies cohérentes en $\mathcal{SHIQ}$. L'ontologie de révision $\mathcal{O}^*$ de $\mathcal{O}$ par $\mathcal{O}'$ est une approximation supérieure de $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$. De plus, la taille de $\mathcal{O}^*$ est bornée par une fonction doublement exponentielle par rapport à la taille de $\mathcal{O}$ et $\mathcal{O}'$.*

Démonstration. Pour montrer que $\mathcal{O}^*$ est une *approximation supérieur* de $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$, nous devons montrer les conditions dans la Définition 14.

(1) Par la construction de $\mathcal{O}^*$, nous avons $\mathcal{S}(\mathcal{O}^*) \subseteq \mathcal{S}(\mathcal{O}) \cup \mathcal{S}(\mathcal{O}')$.

(2) Nous montrons $\mathsf{FM}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O}^*)$.

Soit $\mathsf{FM}(\mathcal{O}^*)$ l'ensemble de modèles de forêt obtenus par l'algorithme de tableau algorithme appliquant à $\mathcal{O}^*$. D'abord, nous démontrons $\mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}') = \mathsf{sub}(\mathcal{O}^*)$. Soit $C \in \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$. Lors de l'application de l'algorithme de tableau sur $\mathcal{O}'$, nous avons $C \in L(x)$ ou $\neg C \in L(x)$ pour chaque nœud x et chaque forêt $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$. Par la construction de $\mathcal{O}^*$, C ou $\neg C$ doit se trouver dans chaque disjoint de la disjonction $\bigsqcup_{1 \leq i \leq n, x \in \mathcal{V}_i} (\bigsqcap_{C \in L(x)} C)$. Cela implique que

$\{C, \neg C\} \subseteq \mathsf{sub}(\mathcal{O}^*)$. Inversement, soit $C \in \mathsf{sub}(\mathcal{O}^*)$. Par la définition de $\mathsf{sub}(\mathcal{O}^*)$, C ou $\neg C$ doit se trouver dans le nouvel axiome de $\mathcal{O}^*$ en tant qu'un sous-concept. Par la construction de $\mathcal{O}^*$, C ou $\neg C$ doit se trouver en tant qu'un sous-concept dans l'étiquette d'un nœud d'un modèle de forêt dans $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$. Selon la Définition 13 concernant $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$, et la Définition 9 concernant $\mathsf{sub}(\mathcal{O})$, il s'ensuit $\{C, \neg C\} \subseteq \mathsf{sub}(\mathcal{O}') \cup \mathsf{sub}(\mathcal{O})$. Donc, $\mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}') = \mathsf{sub}(\mathcal{O}^*)$.

Nous montrons ensuite que $\mathsf{FM}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O}^*)$. L'argument suivant repose sur le fait que lorsque nous connaissons un modèle de forêt $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathcal{O}')$ nous savons également le résultat de l'application de la sat-règle (i.e. le sous-ensemble S choisi à partir de $\mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$) à chaque nœud de $\mathcal{F}$. Cela permet à l'algorithme tableau fonctionnant sur $\mathcal{O}^*$ pour choisir le même S (par la sat-règle) à partir de $\mathsf{sub}(\mathcal{O}^*) = \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$ pour construire un modèle de forêt $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}^*)$. Cet ensemble S est sélectionnable lors de l'application de l'algorithme de tableau sur $\mathcal{O}^*$ car la sat-règle appliquant à un nœud de $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}^*)$ peut toujours choisir une disjoint appropriée du subsumer de l'axiome de concept ajouté à $\mathcal{O}^*$ (*).

Soit $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathcal{O}')$. Par la définition de $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$ il existe une séquence $Seq_{\mathcal{F}}$ d'applications de règle exécutées par l'algorithme de tableau sur $\mathcal{O}'$ avec l'importation d'un ensemble de sous-concepts $\mathsf{sub}(\mathcal{O})$ telle que $Seq_{\mathcal{F}}$ permet de construire $\mathcal{F}$. Afin de montrer que $\mathcal{F} \in \mathsf{FM}(\mathcal{O}^*)$, nous construisons une séquence $Seq_{\mathcal{F}'}$ d'applications de règle exécutées par l'algorithme de tableau sur $\mathcal{O}^*$ telle que $Seq_{\mathcal{F}'}$ permet de construire un modèle de forêt $\mathcal{F}' = \mathcal{F}$. Nous la montrons par induction sur $Seq_{\mathcal{F}}$.

Cas de base. En exécutant l'algorithme de tableau sur $\mathcal{O}'$ avec la $\mathsf{sat}_{\mathcal{R}}$-règle fonctionnant sur $\mathbf{R}_{\mathcal{O}}$ et la sat-règle fonctionnant sur $\mathsf{sub}(\mathcal{O}') \cup \mathsf{sub}(\mathcal{O})$, elle donne une séquence $Seq_{\mathcal{F}}$ telle que les premières applications de règles $R_0, \ldots, R_j$ de $Seq_{\mathcal{F}}$ doivent être celles de la $\mathsf{sat}_{\mathcal{R}}$-règle et donc la sat-règle, et (i) chaque R_l $(0 \leq l \leq k)$ de la $\mathsf{sat}_{\mathcal{R}}$-règle ajoute dans chaque étiquette d'une arête entre des nœuds racines un sous-ensemble éventuellement vide de $\mathbf{R}_{\mathcal{O}}$, (ii) chaque R_i $(k+1 \leq i \leq j)$ de la sat-règle ajoute dans l'étiquette d'un nœud racine $\hat{x}$ deux ensembles S et $\bar{S}$ avec $S \subseteq \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$. Soit $\mathcal{F}_k$ la forêt appliquée jusqu'à la $\mathsf{sat}_{\mathcal{R}}$-règle R_k. Soit $\mathcal{F}_{R_{k+1}}, \ldots, \mathcal{F}_{R_j}$ les modèles de forêt actuels construits en appliquant les règles

$R_{k+1}, \ldots, R_j$ dans $Seq_{\mathcal{F}}$. Dans ce cas, nous ajoutons dans $Seq_{\mathcal{F}'}$ les applications des sat-règles $R'_{k+1}, \ldots, R'_j$ telles que chaque R'_i $(k+1 \leq i \leq j)$ choit les mêmes ensembles S et $\bar{S}$ ci-dessus en raisons de (*). Soit $\mathcal{F}'_k$ la forêt initialisée en utilisant les assertions de $\mathcal{O}^*$ incluant toutes les assertions de $\mathcal{O}'$. Par la construction de $\mathcal{O}^*$, $\mathcal{F}'_k$ et $\mathcal{F}_k$ sont équivalentes. Soit $\mathcal{F}'_{R'_{k+1}}, \ldots, \mathcal{F}'_{R'_j}$ les modèles de forêt actuels construits en appliquant les règles $R'_{k+1}, \ldots, R'_j$ de $Seq_{\mathcal{F}'}$. Nous initialisons un isomorphisme π de $\mathcal{F}'$ à $\mathcal{F}$ tel que $\pi(\widehat{x}') = \widehat{x}$ pour chaque nœud racine $\widehat{x}'$ de $\mathcal{F}'$, et deux modèles de forêt $\mathcal{F}_{R_j}$ et $\mathcal{F}'_{R'_j}$ sont équivalents.

Cas général. Soit $\mathcal{F}_{R_i}$ le modèle de forêt actuel construit en appliquant les règles $R_{k+1}, \cdots, R_i$ de $\{R_{k+1}, \ldots, R_i, R_{i+1}, \ \ldots, R_j\}$ dans $Seq_{\mathcal{F}}$. Soit $x \in \mathcal{F}$ le nœud traité par R_i. Soit $\mathcal{F}'_{R'_i}$ le modèle de forêt actuel construit en appliquant les règles $R'_{k+1}, \cdots, R'_i$ où $Seq_{\mathcal{F}'} = \{R'_{k+1}, \cdots, R'_i, R'_{i+1}, \cdots, R'_j\}$. Soit $x' \in \mathcal{F}'$ le nœud traité par R'_i tel que $\pi(x') = x$, et deux forêts $\mathcal{F}_{R_i}$ et $\mathcal{F}'_{R'_i}$ sont équivalentes.

Puisque deux forêts $\mathcal{F}_{R_i}$ et $\mathcal{F}'_{R'_i}$ sont équivalentes, si une règle différente de $\mathsf{sat}_{\mathcal{R}}$-, sat-, $\exists$- et $\geq$-règles est applicable à un nœud $\pi(y') = y$ de $\mathcal{F}_{R_i}$ alors elle est également applicable à un nœud y' de $\mathcal{F}'_{R'_i}$, et l'isomorphisme π est maintenu. De plus, la $\mathsf{sat}_{\mathcal{R}}$-règle est appliquée sur $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}^*)$. Nous considérons les cas suivants pour la règle R_{i+1} dans $Seq_{\mathcal{F}}$.

(a) R_{i+1} est une sat-règle. En exécutant l'algorithme de tableau sur $\mathcal{O}'$ avec $\mathsf{sub}(\mathcal{O})$, la règle R_{i+1} ajoute dans l'étiquette d'un nœud $\pi(y') = y$ de $\mathcal{F}_{R_i}$ deux ensembles S et $\bar{S}$ tels que $S \subseteq \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$. Puisque π garantit l'équivalence entre $\mathcal{F}_{R_i}$ et $\mathcal{F}'_{R'_i}$, R_{i+1} est aussi applicable sur y' appartenant à $\mathcal{F}'_{R'_i}$ telle que R'_{i+1} ajoute les mêmes ensembles S et $\bar{S}$ (en raison de (*)) à l'étiquette de y'. Nous ajoutons $R'_{i+1} = R_{i+1}$ dans $Seq_{\mathcal{F}'}$. L'isomorphisme π est maintenu.

(b) R_{i+1} est une $\forall$-règle. Cela implique que R_{i+1} est applicable à un concept $\forall S.C$ dans l'étiquette de $\pi(y') = y$ qui appartient à $\mathcal{F}_{R_i}$. Soit $\pi(z') = z$ un S-voisin de $\pi(y')$ tel que R_{i+1} ajoute X dans $L(\pi(z'))$ pour un $X \in \mathsf{Flat}(C)$. Puisque π garantit l'équivalence entre $\mathcal{F}_{R_i}$ et $\mathcal{F}'_{R'_i}$, R_{i+1} est aussi applicable au concept $\forall S.C$ dans l'étiquette de y' appartenant à $\mathcal{F}'_{R'_i}$ telle que R_{i+1} ajoute X à $L'(z')$ sur $\mathcal{F}'_{R'_i}$. Nous ajoutons $R'_{i+1} = R_{i+1}$ dans $Seq_{\mathcal{F}'}$. L'isomorphisme π est maintenu.

(c) R_{i+1} est une $\forall_+$-règle. Cela implique que R_{i+1} est applicable à un concept $\forall S.C$ dans l'étiquette de $\pi(y') = y$ qui appartient à $\mathcal{F}_{R_i}$. Soit R un rôle avec $R \sqsubseteq S$ et $\pi(z') = z$ un S-voisin de $\pi(y')$ tel que R_{i+1} ajoute $\forall R.C$ à $L(\pi(z'))$. Puisque π garantit l'équivalence entre $\mathcal{F}_{R_i}$ et $\mathcal{F}'_{R'_i}$, R_{i+1} est aussi applicable au concept $\forall S.C$ dans l'étiquette de y' appartenant à $\mathcal{F}'_{R'_i}$ telle que R_{i+1} ajoute $\forall R.C$ à $L'(z')$ sur $\mathcal{F}'_{R'_i}$. Nous ajoutons $R'_{i+1} = R_{i+1}$ dans $Seq_{\mathcal{F}'}$. L'isomorphisme π est maintenu.

(d) R_{i+1} est une $\exists$- ou $\geq$-règle. Cela implique que R_{i+1} est applicable sur $\pi(y') = y$ qui appartient à $\mathcal{F}_{R_i}$. Soit $y_1, \cdots, y_m$ les nœuds créés par R_{i+1} dans $\mathcal{F}_{R_{i+1}}$. Puisque π garantit l'équivalence entre $\mathcal{F}_{R_i}$ et $\mathcal{F}'_{R'_i}$, R_{i+1} est aussi applicable on y' appartenant à $\mathcal{F}'_{R'_i}$. Soi $y'_1, \cdots, y'_m$ les nœuds crées par R'_{i+1} dans $\mathcal{F}'_{R'_{i+1}}$. Nous ajoutons $R'_{i+1} = R_{i+1}$ dans $Seq_{\mathcal{F}'}$. Nous étendons π telle que $\pi(y'_i) = y_i$ pour $1 \leq i \leq m$. L'isomorphisme π est maintenu.

(e) R_{i+1} est une $\leq$-règle. Cela implique que R_{i+1} est applicable sur $\pi(y') = y$ qui appartient à $\mathcal{F}_{R_i}$. Soit $\pi(y'_0) = y_0$ le nœud non-racine fusionné à un voisin $\pi(z') = z$ de $\pi(y') = y$ par R_{i+1} dans $\mathcal{F}_{R_{i+1}}$. Puisque π garantit l'équivalence entre $\mathcal{F}_{R_i}$ et $\mathcal{F}'_{R'_i}$, R_{i+1} est aussi applicable on y' appartenant à $\mathcal{F}'_{R'_i}$ et fusionne le nœud y'_0 au voisin z' de y'. Nous ajoutons $R'_{i+1} = R_{i+1}$ dans $Seq_{\mathcal{F}'}$. L'isomorphisme π est maintenu.

(f) R_{i+1} est une $\leq_r$-règle. Cela implique que R_{i+1} est applicable sur $\pi(y') = y$ qui appartient à $\mathcal{F}_{R_i}$ tandis que la $\leq$-règle n'est pas applicable sur $\pi(y') = y$. Soit $\pi(y'_0) = y_0$ le nœud racine fusionné à un voisin $\pi(z') = z$ de $\pi(y') = y$ par R_{i+1} dans $\mathcal{F}_{R_{i+1}}$. Puisque π garantit l'équivalence entre $\mathcal{F}_{R_i}$ et $\mathcal{F}'_{R'_i}$, la $\leq$-règle n'est pas applicable sur y' et R_{i+1} est applicable sur y' appartenant à $\mathcal{F}'_{R'_i}$ et fusionne le nœud y'_0 au voisin z' de y'. Nous ajoutons $R'_{i+1} = R_{i+1}$ dans $Seq_{\mathcal{F}'}$. L'isomorphisme π est maintenu.

Dans tous les cas de R_{i+1}, l'isomorphisme π est maintenu et deux forêts $\mathcal{F}_{R_{i+1}}$ et $\mathcal{F}'_{R'_{i+1}}$ sont équivalentes. Donc, la séquence $Seq_{\mathcal{F}'}$ d'applications de règle exécutées par l'algorithme de tableau sur $\mathcal{O}'$ avec $\mathsf{sub}(\mathcal{O})$ construit un modèle de forêt $\mathcal{F}' = \mathcal{F}$. Alors, $\mathsf{FM}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O}^*)$.

(3) Nous démontrons qu'il n'existe aucune ontologie $\mathcal{O}''$ telle que $\mathsf{FM}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O}'') \subset \mathsf{FM}(\mathcal{O}^*)$. Par équivalence, nous devons montrer que s'il existe une ontologie $\mathcal{O}''$ telle que $\mathsf{FM}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O}'') \subseteq \mathsf{FM}(\mathcal{O}^*)$ alors $\mathsf{FM}(\mathcal{O}'') = \mathsf{FM}(\mathcal{O}^*)$.

D'abord, nous montrons que $\mathsf{sub}(\mathcal{O}'') = \mathsf{sub}(\mathcal{O}^*)$ en utilisant le comportement de la sat-règle. Soient $C \in \mathsf{sub}(\mathcal{O}'')$ et $\mathcal{F} \in \mathsf{FM}(\mathcal{O}'') \subseteq \mathsf{FM}(\mathcal{O}^*)$. nous avons $C' \in L(x)$ où $C' \in \{C, \neg C\}$ pour chaque nœud x de $\mathcal{F}$ à cause de la sat-règle. Puisque $C' \in L(x)$ et $\mathcal{F} \in \mathsf{FM}(\mathcal{O}^*)$ nous avons $C' \in \mathsf{sub}(\mathcal{O}^*)$ et donc $C \in \mathsf{sub}(\mathcal{O}^*)$ à cause de la sat-règle. Donc, $\mathsf{sub}(\mathcal{O}'') \subseteq \mathsf{sub}(\mathcal{O}^*)$. Nous prouvons maintenant que $\mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}') \subseteq \mathsf{sub}(\mathcal{O}'')$. Soient $C \in \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}')$ et $\mathcal{F} \in \mathsf{FM}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O}'')$. nous avons $C' \in L(x)$ où $C' \in \{C, \neg C\}$ pour chaque nœud x de $\mathcal{F}$ à cause de la sat-règle. Donc, $\mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}') \subseteq \mathsf{sub}(\mathcal{O}'')$. De plus, nous avons montré dans (2) que $\mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}(\mathcal{O}') = \mathsf{sub}(\mathcal{O}^*)$. Alors, $\mathsf{sub}(\mathcal{O}'') = \mathsf{sub}(\mathcal{O}^*)$.

Nous montrons $\mathsf{FM}(\mathcal{O}'') = \mathsf{FM}(\mathcal{O}^*)$. D'abord, nous devons démontrer l'affirmation suivante :

Claim 1. Soit $\mathcal{F} \in \mathsf{FM}(\mathcal{O}^*)$. Chaque sous-ensemble $S \subseteq \mathsf{sub}(\mathcal{O}^*)$ choisi lors de l'application de la sat-règle à un nœud x de $\mathcal{F}$ peut également être choisi lors de l'application de la sat-règle à un nœud x' d'un modèle de forêt $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}'')$.

Soit $\mathcal{F} \in \mathsf{FM}(\mathcal{O}^*)$. Par la construction de $\mathcal{O}^*$ dans la Définition 15 et $\mathcal{F} \in \mathsf{FM}(\mathcal{O}^*)$, l'étiquette de chaque nœud x de $\mathcal{F}$ est l'ensemble des conjoints d'une conjonction qui est une disjoint de la disjonction $\bigsqcup_{1 \leq i \leq n, x \in \mathcal{V}_i} (\bigsqcap_{C \in L(x)} C)$. De plus, cette disjonction est construite à partir des étiquettes de nœud de toute $\mathcal{F}_j \in \mathsf{M}(\mathcal{O}, \mathcal{O}')$ (par la Définition 15). Cela implique que l'étiquette de chaque nœud x de $\mathcal{F}$ est égale à celle d'un nœud x' appartenant à un modèle de forêt $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}, \mathcal{O}')$ (*).

Puisque $\mathsf{FM}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O}'')$, nous pouvons toujours choisir un sous-ensemble $S \subseteq \mathsf{sub}(\mathcal{O}'')$ qui est égal à l'étiquette d'un nœud x' appartenant à un modèle de forêt $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}, \mathcal{O}')$ tel que S est non-clash et satisfait tous les axiomes dans $\mathcal{O}''$ (**).

En raison de (*), (**) et $\mathsf{sub}(\mathcal{O}'') = \mathsf{sub}(\mathcal{O}^*)$, lors de l'application de la sat-règle à un nœud x' d'une forêt $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}'')$, nous pouvons choisir un sous-ensemble $S \subseteq \mathsf{sub}(\mathcal{O}'')$ qui est égal à l'étiquette d'un nœud x de $\mathcal{F}$ tel que S est non-clash et satisfait tous les axiomes dans $\mathcal{O}''$. Alors, la Affirmation 1 est prouvée.

L'argument suivant repose sur le fait que lorsque nous savons un modèle de forêt $\mathcal{F} \in \mathsf{FM}(\mathcal{O}^*)$ nous avons aussi le résultat d'application de la sat-règle, i.e. le sous-ensemble S choisi de $\mathsf{sub}(\mathcal{O}^*)$, à chaque nœud de $\mathcal{F}$. Ceci permet à l'algorithme de tableau exécutant sur $\mathcal{O}''$ de choisir le même S (par la sat-règle) de $\mathsf{sub}(\mathcal{O}'') = \mathsf{sub}(\mathcal{O}^*)$ pour construire un modèle de forêt $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}'')$. Cet ensemble S est sélectionnable lors de l'application de l'algorithme de tableau sur $\mathcal{O}''$ à cause de la Affirmation 1.

Soit $\mathcal{F} \in \mathsf{FM}(\mathcal{O}^*)$. Notons que la $\mathsf{sat}_{\mathcal{R}}$-règle n'est pas appliquée sur $\mathcal{F} \in \mathsf{FM}(\mathcal{O}^*)$. Il existe une séquence $Seq_{\mathcal{F}}$ d'applications de règle exécutées par l'algorithme de tableau sur $\mathcal{O}^*$ telle que $Seq_{\mathcal{F}}$ permet de construire $\mathcal{F} \in \mathsf{FM}(\mathcal{O}^*)$. Pour démontrer que $\mathcal{F} \in \mathsf{FM}(\mathcal{O}'')$, nous construisons une séquence $Seq_{\mathcal{F}'}$ d'applications de règle exécutées par l'algorithme de tableau sur $\mathcal{O}''$ telle que $Seq_{\mathcal{F}'}$ permet de construire une forêt $\mathcal{F}' = \mathcal{F}$. Nous la montrons par induction sur $Seq_{\mathcal{F}}$.

Cas de base. Puisque $\mathsf{FM}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O}'') \subseteq \mathsf{FM}(\mathcal{O}^*)$, alors l'ensemble de noms d'individu de $\mathcal{O}^*$ est aussi celui de $\mathcal{O}''$. En exécutant l'algorithme de tableau sur $\mathcal{O}^*$, nous pouvons choisir une $Seq_{\mathcal{F}}$ telle que les premières applications de règle $R_0, \ldots, R_j$ de $Seq_{\mathcal{F}}$ doivent être celles de la sat-règle, et chaque R_i $(0 \leq i \leq j)$ ajoute dans l'étiquette d'un nœud racine $\widehat{x}$ deux ensembles S et $\bar{S}$ avec $S \subseteq \mathsf{sub}(\mathcal{O}^*)$. Soit $\mathcal{F}_0$ la forêt initialisée en utilisant les assertions de $\mathcal{O}^*$ comme l' initialisation des autres forêts $\mathcal{F}_i \in \mathsf{FM}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O}^*)$. Soit $\mathcal{F}_{R_0}, \ldots, \mathcal{F}_{R_j}$ les forêts actuelles construites en appliquant les règles $R_0, \ldots, R_j$ dans $Seq_{\mathcal{F}}$. Dans ce cas, nous ajoutons dans les applications de la $Seq_{\mathcal{F}'}$ sat-règle $R'_0, \ldots, R'_j$ telles que chaque R'_i $(0 \leq i \leq j)$ choisit les mêmes ensembles S et $\bar{S}$ ci-dessus à cause de la Affirmation 1. Soit $\mathcal{F}'_0$ la forêt initialisée en utilisant les assertions de $\mathcal{O}''$ comme l'initialisation des autres forêts $\mathcal{F}'_i \in \mathsf{FM}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{FM}(\mathcal{O}'')$. Soit $\mathcal{F}'_{R'_0}, \ldots, \mathcal{F}'_{R'_j}$ les forêts actuelles construites en appliquant les règles $R'_0, \ldots, R'_j$ de $Seq_{\mathcal{F}'}$. Nous initialisons un isomorphisme π de $\mathcal{F}'$ à $\mathcal{F}$ tel que $\pi(\widehat{x}') = \widehat{x}$ pour chaque nœud racine $\widehat{x}'$ de $\mathcal{F}'$, et deux forêts $\mathcal{F}_{R_j}$ et $\mathcal{F}'_{R'_j}$ sont équivalentes.

Cas général. Soit $\mathcal{F}_{R_i}$ la forêt actuelle construite par les applications de règles $R_1, \cdots, R_i$ où $Seq_{\mathcal{F}} = \{R_1, \ldots, R_i, R_{i+1}, \ldots, R_k\}$. Soit $x \in \mathcal{F}$ le nœud traité par R_i. Let $\mathcal{F}'_{R'_i}$ la forêt actuelle construite par les applications de règles $R'_1, \cdots, R'_i$ où

$Seq_{\mathcal{F}'} = \{R'_1, \cdots, R'_i, R'_{i+1}, \cdots, R'_k\}$. Soit $x' \in \mathcal{F}'$ le nœud traité par R'_i tel que $\pi(x') = x$, et deux forêts $\mathcal{F}_{R_i}$ et $\mathcal{F}'_{R'_i}$ sont équivalentes.

Puisque deux forêts $\mathcal{F}_{R_i}$ et $\mathcal{F}'_{R'_i}$ sont équivalentes, si une règle différente de la $\mathsf{sat}_{\mathcal{R}}$-, sat-, $\exists$- et $\geq$-règles est applicable à un nœud $\pi(y') = y$ de $\mathcal{F}_{R_i}$ alors elle est aussi applicable à un nœud y' de $\mathcal{F}'_{R'_i}$ et l'isomorphisme π est maintenu. De plus, la $\mathsf{sat}_{\mathcal{R}}$-règle n'est pas appliquée sur $\mathcal{F}' \in \mathsf{FM}(\mathcal{O}'')$. Nous considérons les cas suivants de la règle R_{i+1} dans $Seq_{\mathcal{F}}$.

(a) R_{i+1} est une sat-règle. En exécutant l'algorithme de tableau sur $\mathcal{O}^*$, la règle R_{i+1} ajoute dans l'étiquette d'un nœud $\pi(y') = y$ de $\mathcal{F}_{R_i}$ deux ensembles S et $\bar{S}$ tels que $S \subseteq \mathsf{sub}(\mathcal{O}^*)$. Puisque π garantit l'équivalence entre $\mathcal{F}_{R_i}$ et $\mathcal{F}'_{R'_i}$, R_{i+1} est aussi applicable sur y' appartenant à $\mathcal{F}'_{R'_i}$ telle que R'_{i+1} ajoute les mêmes ensembles S et $\bar{S}$ (à cause de la Affirmation 1) à l'étiquette de y'. Nous ajoutons $R'_{i+1} = R_{i+1}$ dans $Seq_{\mathcal{F}'}$. L'isomorphisme π est maintenu.

(b) R_{i+1} est une $\forall$-règle. Cela implique que R_{i+1} est applicable à un concept $\forall S.C$ dans l'étiquette de $\pi(y') = y$ qui appartient à $\mathcal{F}_{R_i}$. Soit $\pi(z') = z$ un S-voisin de $\pi(y')$ tel que R_{i+1} ajoute X dans $L(\pi(z'))$ pour un $X \in \mathsf{Flat}(C)$. Puisque π garantit l'équivalence entre $\mathcal{F}_{R_i}$ et $\mathcal{F}'_{R'_i}$, R_{i+1} est aussi applicable au concept $\forall S.C$ dans l'étiquette de y' appartenant à $\mathcal{F}'_{R'_i}$ telle que R_{i+1} ajoute X dans $L'(z')$ sur $\mathcal{F}'_{R'_i}$. Nous ajoutons $R'_{i+1} = R_{i+1}$ dans $Seq_{\mathcal{F}'}$. L'isomorphisme π est maintenu.

(c) R_{i+1} est une $\forall_+$-règle. Cela implique que R_{i+1} est applicable à un concept $\forall S.C$ dans l'étiquette de $\pi(y') = y$ qui appartient à $\mathcal{F}_{R_i}$. Soient R un rôle avec $R \boxplus S$ et $\pi(z') = z$ un S-voisin de $\pi(y')$ tels que R_{i+1} ajoute $\forall R.C$ dans $L(\pi(z'))$. Puisque π garantit l'équivalence entre $\mathcal{F}_{R_i}$ et $\mathcal{F}'_{R'_i}$, R_{i+1} est aussi applicable au concept $\forall S.C$ dans l'étiquette de y' appartenant à $\mathcal{F}'_{R'_i}$ telle que R_{i+1} ajoute $\forall R.C$ dans $L'(z')$ sur $\mathcal{F}'_{R'_i}$. Nous ajoutons $R'_{i+1} = R_{i+1}$ dans $Seq_{\mathcal{F}'}$. L'isomorphisme π est maintenu.

(d) R_{i+1} est une $\exists$- ou $\geq$-règle. Cela implique que R_{i+1} est applicable sur $\pi(y') = y$ qui appartient à $\mathcal{F}_{R_i}$. Soit $y_1, \cdots, y_m$ les nœuds crées par R_{i+1} dans $\mathcal{F}_{R_{i+1}}$. Puisque π garantit l'équivalence entre $\mathcal{F}_{R_i}$ et $\mathcal{F}'_{R'_i}$, R_{i+1} est aussi applicable on

y' appartenant à $\mathcal{F}'_{R'_i}$. Soit $y'_1, \cdots, y'_m$ les nœuds crées par R'_{i+1} dans $\mathcal{F}'_{R'_{i+1}}$. Nous ajoutons $R'_{i+1} = R_{i+1}$ dans $Seq_{\mathcal{F}'}$. Nous étendons π telle que $\pi(y'_i) = y_i$ pour $1 \leq i \leq m$. L'isomorphisme π est maintenu.

(e) R_{i+1} est une $\leq$-règle. Cela implique que R_{i+1} est applicable sur $\pi(y') = y$ qui appartient à $\mathcal{F}_{R_i}$. Soit $\pi(y'_0) = y_0$ le nœud racine fusionné à un voisin $\pi(z') = z$ de $\pi(y') = y$ par R_{i+1} dans $T_{R_{i+1}}$. Puisque π garantit l'équivalence entre $\mathcal{F}_{R_i}$ et $\mathcal{F}'_{R'_i}$, R_{i+1} est aussi applicable sur y' appartenant à $\mathcal{F}'_{R'_i}$ et fusionne le nœud y'_0 au voisin z' de y'. Nous ajoutons $R'_{i+1} = R_{i+1}$ dans $Seq_{\mathcal{F}'}$. L'isomorphisme π est maintenu.

(f) R_{i+1} est une $\leq_r$-règle. Cela implique que R_{i+1} est applicable sur $\pi(y') = y$ qui appartient à $\mathcal{F}_{R_i}$ tandis que la $\leq$-règle n'est pas applicable sur $\pi(y') = y$. Soit $\pi(y'_0) = y_0$ le nœud racine fusionné à un autre nœud racine étant un voisin $\pi(z') = z$ de $\pi(y') = y$ par R_{i+1} dans $\mathcal{F}_{R_{i+1}}$. Puisque π garantit l'équivalence entre $\mathcal{F}_{R_i}$ et $\mathcal{F}'_{R'_i}$, la $\leq$-règle n'est pas applicable sur y' et R_{i+1} est applicable sur y' appartenant à $\mathcal{F}'_{R'_i}$ et fusionne le nœud y'_0 au voisin z' de y'. Nous ajoutons $R'_{i+1} = R_{i+1}$ dans $Seq_{\mathcal{F}'}$. L'isomorphisme π est maintenu.

Dans tous les cas de R_{i+1}, l'isomorphisme π est maintenu et deux forêts $\mathcal{F}_{R_{i+1}}$ et $\mathcal{F}'_{R'_{i+1}}$ sont équivalentes. Donc, la séquence $Seq_{\mathcal{F}'}$ d'applications de règle exécutées par l'algorithme de tableau sur $\mathcal{O}''$ construit un modèle de forêt $\mathcal{F}' = \mathcal{F}$. Alors, $\mathsf{FM}(\mathcal{O}^*) \subseteq \mathsf{FM}(\mathcal{O}'')$. Avec $\mathsf{FM}(\mathcal{O}'') \subseteq \mathsf{FM}(\mathcal{O}^*)$, nous avons $\mathsf{FM}(\mathcal{O}'') = \mathsf{FM}(\mathcal{O}^*)$.

□

Lemme 8. *Une valeur $v_k \in \Delta$ est la distance entre $T\langle x_0\rangle$ et $T\langle z_0\rangle$ si et seulement si,*

(i) il existe un isomorphisme π et un lien $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$ tels que $d(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle) = v_k$ et $d(\langle w, z\rangle, \langle \pi(w), \pi(z)\rangle) \leq v_k$ pour tous les liens $(\langle w, z\rangle, \langle \pi(w), \pi(z)\rangle)$ différents de $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$, et

(ii) pour chaque isomorphisme π, il existe un lien $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$ tel que $d(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle) \geq v_k$.

Démonstration. Par la construction, la distance entre $T\langle x_0\rangle$ et $T\langle z_0\rangle$ est un des valeurs dans l'ensemble Δ.

• Nous prouvons la direction "si" : Pour ce faire, nous montrons que si le Point *(i)* et le Point *(ii)* sont satisfaits alors v_k est la distance entre $T\langle x_0\rangle$ et $T\langle z_0\rangle$.

1. Soit π_0 un isomorphisme entre $T\langle x_0\rangle$ et $T\langle z_0\rangle$ qui a un lien $(\langle x,y\rangle, \langle \pi(x), \pi(y)\rangle)$ tel que $d(\langle x,y\rangle, \langle \pi(x), \pi(y)\rangle) = v_k$ et $d(\langle w,z\rangle, \langle \pi(w), \pi(z)\rangle) \leq v_k$ pour tous les liens $(\langle w,z\rangle, \langle \pi(w), \pi(z)\rangle)$ (hypothèses).
2. Pour tout π, il existe un lien $(\langle x,y\rangle, \langle \pi(x), \pi(y)\rangle)$ tel que $d(\langle x,y\rangle, \langle \pi(x), \pi(y)\rangle) \geq v_k$ (hypothèses).
3. $\max\limits_{\langle w,z\rangle \in E_1} (|L_1(w) \vartriangle L_2(\pi_0(w))|$
   $+|L_1(\langle w,z\rangle) \vartriangle L_2(\langle \pi_0(w), \pi_0(z)\rangle)|$
   $+|L_1(z) \vartriangle L_2(\pi_0(z))|) = v_k$ (par le Point 1)
4. Pour chaque π entre $T\langle x_0\rangle$ et $T\langle z_0\rangle$, $\max\limits_{\langle w,z\rangle \in E_1} (|L_1(w) \vartriangle L_2(\pi(w))|$
   $+|L_1(\langle w,z\rangle) \vartriangle L_2(\langle \pi(w), \pi(z)\rangle)|$
   $+|L_1(z) \vartriangle L_2(\pi(z))|) \geq v_k$ (par le Point 2)
5. $d(T\langle x_0\rangle, T\langle z_0\rangle) = v_k$ (par le Point 3 et le Point 4).

• Nous prouvons la direction "seulement si". Nous devons montrer que si v_k est la distance entre $T\langle x_0\rangle$ et $T\langle z_0\rangle$ alors le Point *(i)* et le Point *(ii)* sont satisfaits.

1. Supposons que v_k est la distance entre $T\langle x_0\rangle$ et $T\langle z_0\rangle$.
2. il existe un π_0 tel que $\max\limits_{\langle w,z\rangle \in E_1} (|L_1(w) \vartriangle L_2(\pi_0(w))|$
   $+|L_1(\langle w,z\rangle) \vartriangle L_2(\langle \pi_0(w), \pi_0(z)\rangle)|$
   $+|L_1(z) \vartriangle L_2(\pi_0(z))|) = v_k$ (par le Point 1)
3. il existe un lien $(\langle x,y\rangle, \langle \pi_0(x), \pi_0(y)\rangle)$ tel que $d(\langle x,y\rangle, \langle \pi_0(x), \pi_0(y)\rangle) = v_k$ et $d(\langle w,z\rangle, \langle \pi_0(w), \pi_0(z)\rangle) \leq v_k$ pour tous les liens $(\langle w,z\rangle, \langle \pi_0(w), \pi_0(z)\rangle)$ (par le Point 2).
4. pour chaque π, nous avons $\max\limits_{\langle w,z\rangle \in E_1} (|L_1(w) \vartriangle L_2(\pi(w))|$
   $+|L_1(\langle w,z\rangle) \vartriangle L_2(\langle \pi(w), \pi(z)\rangle)|$
   $+|L_1(z) \vartriangle L_2(\pi(z))|) \geq v_k$ (par le Point 1)
5. pour chaque isomorphisme π, il existe un lien $(\langle x,y\rangle, \langle \pi(x), \pi(y)\rangle)$ tel que $d(\langle x,y\rangle, \langle \pi(x), \pi(y)\rangle) \geq v_k$ (par le Point 4).

6. le Point *(i)* et le Point *(ii)* dans le Lemme 8 sont prouvés à cause du Point 3 et le Point 5.

□

Avant de prouver le Lemme 9, nous devons montrer les lemmes suivants.

Lemme 10. *Soit $\bar{v} \in \Delta$. Si l'Algorithme 3 retourne une valeur $\widehat{v} \in \Delta$ telle que $\widehat{v} < \bar{v}$ alors il existe un isomorphisme π tel que $d(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle) < \bar{v}$ pour tous les liens $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$.*

Démonstration.

1. Soit $\bar{v} \in \Delta$. Supposons que l'Algorithme 3 retourne par la Ligne 18 une valeur $\widehat{v} \in \Delta$ telle que $\widehat{v} < \bar{v}$. Cela implique que, dans la boucle de la Ligne 4 dans l'Algorithme 3, il y a une itération correspondant à $\bar{v}$ telle qu'il existe un ensemble $\mathsf{LINKS}(\mathsf{succ}(x_0), \mathsf{succ}(z_0))$ dont chaque lien est vert (par la Ligne 17).
2. Initialisons une fonction π qui contient tous les liens verts de l'ensemble $\mathsf{LINKS}(\mathsf{succ}(x_0), \mathsf{succ}(z_0))$ (indiqué dans le Point 1).
3. Au niveau $n = 1$ dans les arbres, pour chaque lien vert $(\langle x_0, y\rangle, \langle z_0, w\rangle)$ (indiqué dans dans le Point 2), il existe un ensemble $\mathsf{LINKS}(\mathsf{succ}(y), \mathsf{succ}(w))$ dont chaque lien est vert (par la Ligne 10 et la Ligne 11 de l'Algorithme 3).
4. π est étendu en ajoutant tous les liens verts de l'ensemble $\mathsf{LINKS}(\mathsf{succ}(y), \mathsf{succ}(w))$ indiqué dans le Point 3.
5. Supposons que à chaque niveau $2 \leq n \leq h-1$, pour chaque lien vert $(\langle y, y'\rangle, \langle w, w'\rangle)$ au niveau n, il existe un ensemble $\mathsf{LINKS}(\mathsf{succ}(y'), \mathsf{succ}(w'))$ dont chaque lien est vert. En particulier, au niveau $n = k$, pour chaque lien vert $(\langle y, y'\rangle, \langle w, w'\rangle)$ au niveau k, il existe un ensemble $\mathsf{LINKS}(\mathsf{succ}(y'), \mathsf{succ}(w'))$ dont chaque lien est vert (par la Ligne 10 et la Ligne 11 de l'Algorithme 3).
6. π est étendu en ajoutant tous les liens verts de l'ensemble $\mathsf{LINKS}(\mathsf{succ}(y'), \mathsf{succ}(w'))$ indiqué dans le Point 5.
7. À la itération correspondant à $\bar{v}$ dans la boucle de la Ligne 4 de l'Algorithme 3, π passe seulement par des liens verts (par les Points 1, 2, 4 et 6). Quand la boucle se termine , π devient un isomorphisme.

8. π ne passe pas par aucun lien dans $\mathsf{COOR}(\bar{v})$ (par le Point 7 et la Ligne 5 de l'Algorithme 3).
9. $d(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle) < \bar{v}$ pour tous les liens $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$ (par le Point 8).

□

Lemme 11. *Soit $\widehat{v} \in \Delta$. Si l'Algorithme 3 retourne la valeur $\widehat{v}$, alors*

(i) il existe un isomorphisme π et un lien $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$ tels que $d(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle) = \widehat{v}$ et $d(\langle w, z\rangle, \langle \pi(w), \pi(z)\rangle) \leq \widehat{v}$ pour tous les liens $(\langle w, z\rangle, \langle \pi(w), \pi(z)\rangle)$, et

(ii) pour tous les isomorphismes π, il existe un lien $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$ tel que $d(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle) \geq \widehat{v}$.

Démonstration.

1. Supposons que l'Algorithme 3 retourne $\widehat{v} \in \Delta$. Chaque isomorphisme π doit passer par un ensemble $\mathsf{LINKS}(\mathsf{succ}(x_0), \mathsf{succ}(z_0))$ dont chaque lien est rouge (par la Ligne 17 et la Ligne 18 de l'Algorithme 3).
2. Chaque isomorphisme π doit passer par un lien $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle) \in \mathsf{COOR}(\bar{v})$ pour toute $\bar{v} \in \Delta$ avec $\bar{v} \geq \widehat{v}$ (par le Point 1, la Ligne 5, la Ligne 8 et la Ligne 9 de l'Algorithme 3).
3. Le Point *(ii)* dans le Lemme 11 est prouvé à cause du Point 2.
4. Nous avons que l'Algorithme 3 ne retourne pas la valeur $\widehat{v} + 1$ (par le Point 1).
   i. Il existe un isomorphisme π_0 tel que $d(\langle x, y\rangle, \langle \pi_0(x), \pi_0(y)\rangle) < \widehat{v} + 1$ pour tous les liens $(\langle x, y\rangle, \langle \pi_0(x), \pi_0(y)\rangle)$ (par le Point 1 et le Lemme 10).
   ii. Pour l'isomorphisme π_0 indiqué dans le Point i., il y a un lien $(\langle x, y\rangle, \langle \pi_0(x), \pi_0(y)\rangle)$ avec $d(\langle x, y\rangle, \langle \pi_0(x), \pi_0(y)\rangle) = \widehat{v}$ et $d(\langle w, z\rangle, \langle \pi_0(w), \pi_0(z)\rangle) \leq \widehat{v}$ pour tous les liens $(\langle w, z\rangle, \langle \pi_0(w), \pi_0(z)\rangle)$ (par les Points 2 et i.).
   iii. Le Point *(i)* dans le Lemme 11 est prouvé (par le Point ii.).

□

Lemme 12. *Soit $\widehat{v} \in \Delta$. L'Algorithme 3 retourne la valeur $\widehat{v}$ si*

(i) il existe un isomorphisme π *et un lien* $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$ *tels que* $d(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle) = \widehat{v}$ *et* $d(\langle w, z\rangle, \langle \pi(w), \pi(z)\rangle) \leq \widehat{v}$ *pour tous les liens* $(\langle w, z\rangle, \langle \pi(w), \pi(z)\rangle)$, *et*

(ii) pour tout isomorphisme π, *il existe un lien* $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$ *tel que* $d(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle) \geq \widehat{v}$.

Démonstration. 1. Soit $\widehat{v} \in \Delta$. Pour tout isomorphisme π, il existe un lien $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$ tel que $d(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle) \geq \widehat{v}$ (hypothèses).

2. L'Algorithme 3 ne retourne pas une valeur $v \in \Delta$ telle que $v < \widehat{v}$ (par le Point 1 et le Lemme 10).

3. L'Algorithme 3 retourne une valeur $\bar{v} \in \Delta$ telle que $\bar{v} \geq \widehat{v}$ (par le Point 2).

4. Il existe un isomorphisme π_0 et un lien $(\langle x, y\rangle, \langle \pi_0(x), \pi_0(y)\rangle)$ tels que $d(\langle x, y\rangle, \langle \pi_0(x), \pi_0(y)\rangle) = \widehat{v}$ et $d(\langle w, z\rangle, \langle \pi_0(w), \pi_0(z)\rangle) \leq \widehat{v}$ pour tous les liens $(\langle w, z\rangle, \langle \pi_0(w), \pi_0(z)\rangle)$ (hypothèses).

5. À chaque itération dans la boucle de la Ligne 4 dans l'Algorithme 3, correspondant à chaque valeur $v_k \in \Delta$ avec $v_k > \widehat{v}$, il existe toujours l'isomorphisme π_0 indiqué dans le Point 4 tel que $d(\langle x, y\rangle, \langle \pi_0(x), \pi_0(y)\rangle) < v_k$ pour tous les liens $(\langle x, y\rangle, \langle \pi_0(x), \pi_0(y)\rangle)$ (par le Point 4 et l'Algorithme 3). Cela implique que, pour chaque itération correspondant à la valeur $v_k > \widehat{v}$, l'isomorphisme π_0 passe seulement par des liens verts.

6. L'Algorithme 3 ne retourne pas la valeur v_k où $v_k > \widehat{v}$ (par le Point 5 et la Ligne 17 de l'Algorithme 3).

7. L'Algorithme 3 retourne la valeur $\widehat{v}$ (par le Point 3 et le Point 6).

□

Nous pouvons maintenant prouver le Lemme 9.

Lemme 9. *1. Soient* $T\langle x_0\rangle = (V_1, E_1, L_1)$, $T\langle z_0\rangle = (V_2, E_2, L_2)$ *deux arbres. L'algorithme 3 retourne la distance entre* $T\langle x_0\rangle$ *et* $T\langle z_0\rangle$.

2. L'algorithme 3 s'exécute en temps polynomial en fonction de la taille de $T\langle x_0\rangle$ *et de* $T\langle z_0\rangle$.

Démonstration. 1. Soient $T\langle x_0\rangle = (V_1, E_1, L_1)$, $T\langle z_0\rangle = (V_2, E_2, L_2)$ deux arbres. Nous prouvons que :

(a) si l'Algorithme 3 retourne une valeur $\widehat{v}$, alors $\widehat{v}$ est la distance entre $T\langle x_0\rangle$ et $T\langle z_0\rangle$;

(b) si une valeur $\widehat{v}$ est la distance entre $T\langle x_0\rangle$ et $T\langle z_0\rangle$, l'Algorithme 3 retourne $\widehat{v}$.

• Nous montrons (a) :

i. L'Algorithme 3 retourne une valeur $\widehat{v}$ (hypothèses)

ii. Il existe un isomorphisme π et un lien $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$ tels que $d(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle) = \widehat{v}$ et $d(\langle w, z\rangle, \langle \pi(w), \pi(z)\rangle) \leq \widehat{v}$ pour tous les liens $(\langle w, z\rangle, \langle \pi(w), \pi(z)\rangle)$, et pour tout π, il existe un lien $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$ tel que $d(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle) \geq \widehat{v}$ (par le Point i. et le Lemme 11).

iii. $\widehat{v}$ est la distance entre $T\langle x_0\rangle$ et $T\langle z_0\rangle$ (par le Point ii. et le Lemme 8).

• Nous prouvons (b) :

i. $\widehat{v}$ est la distance entre $T\langle x_0\rangle$ et $T\langle z_0\rangle$ (hypothèses).

ii. Il existe un isomorphisme π et un lien $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$ tels que $d(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle) = \widehat{v}$ et $d(\langle w, z\rangle, \langle \pi(w), \pi(z)\rangle) \leq \widehat{v}$ pour tous les liens $(\langle w, z\rangle, \langle \pi(w), \pi(z)\rangle)$, et pour tout π, il existe un lien $(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle)$ tel que $d(\langle x, y\rangle, \langle \pi(x), \pi(y)\rangle) \geq \widehat{v}$ (par le Point i. et le Lemme 12).

iii. L'Algorithme 3 retourne une valeur $\widehat{v}$ (par le Point ii. et le Lemme 8).

2. Nous devons montrer le temps d'exécution de chaque étape de l'Algorithme 3 est polynomial. Pour ce faire, supposons que V est l'ensemble de nœuds et E l'ensemble d'arêtes sur un arbre. Puisque chaque arbre est complet, nous avons la hauteur de l'arbre $h = \mathsf{log}_n(\mathsf{LN})$ où n est le nombre maximal de successeurs d'un nœud sur l'arbre complet, LN est le nombre de nœuds feuilles. Cela implique que $h \leq \mathsf{log}_n|V|$. Donc, nous avons au plus $h = \mathsf{log}_n|V|$.

• Nous considérons la Ligne 3 de l'Algorithme 3 : Pour chaque valeur v_k de Δ, il est nécessaire d'exécuter au plus $|E|\times|E|$ itérations pour trouver un ensemble $\mathsf{COOR}(v_k)$. Donc, le temps d'exécution de la Ligne 3 de l'Algorithme 3 est $O(|\Delta| \times |E|^2)$.

• Nous considérons la Ligne 5 de l'Algorithme 3 : Le nombre de liens dans chaque $\mathsf{COOR}(v_k)$ est au plus $|E|$. Donc, le temps pour colorier en rouge à tous les liens dans $\mathsf{COOR}(v_k)$ est au plus $|E|$. Alors, le temps d'exécution de la Ligne 5 de l'Algorithme 3 est $O(|E|)$.

• Nous considérons les Lignes 6-18 de l'Algorithme 3 :

i. Pour chaque lien $(\langle x, y\rangle, \langle z, w\rangle)$,l'algorithme doit exécuter au plus $h-1$ itérations pour vérifier s'il y a un lien rouge $(\langle x', y'\rangle, \langle z', w'\rangle)$ tel que y' (resp. w') est un ancêtre de y (resp. w) car $\langle x, y\rangle$ et $\langle x', y'\rangle$ (resp. $\langle z, w\rangle$ et $\langle z', w'\rangle$) est sur le chemin de x à y' (resp. de z à w').

ii. Pour chaque lien $(\langle x, y\rangle, \langle z, w\rangle)$, nous utilisons l'algorithme de Hopcroft-Karp [Hopcroft and Karp, 1973] pour trouver un ensemble $\mathsf{LINKS}(\mathsf{succ}(y), \mathsf{succ}(w))$ dont chaque lien est vert. Selon Hopcroft et Karp [Hopcroft and Karp, 1973], la complexité de l'algorithme est $n^2 \times \sqrt{n}$ dans le pire de cas. Cet algorithme prend en entrée un graphe de bipartite (deux ensembles de successeurs) et produit en sortie une correspondance maximale de cardinalité (un ensemble du plus grand nombre possible d'arêtes avec la propriété telle que deux arêtes partagent un point final).

iii. À chaque niveau i, l'algorithme doit vérifier et colorier au plus $|E| \times |E|$ liens $(\langle x, y\rangle, \langle z, w\rangle)$ (car le nombre maximal d'arêtes est $|E|$).

iv. Le temps de vérification et de coloriage pour tous les liens $(\langle x, y\rangle, \langle z, w\rangle)$ au niveau i (indiqué dans le Point iii.) est au plus $O(|E| \times |E| \times ((h-1) + n^2 \times \sqrt{n}))$ (par les Points i., ii. et iii.).

v. Le temps de vérification et de coloriage pour tous les liens sur un arbre est au plus $O(h \times |E|^2 \times (h + n^2 \times \sqrt{n}))$ (par le Point iv.). Alors, le temps d'exécution des Lignes 6-18 de l'Algorithme 3 est $O(\mathsf{log}_n|V| \times |E|^2 \times (\mathsf{log}_n|V| + n^2 \times \sqrt{n}))$.

• Nous considérons la Ligne 17 de l'Algorithme 3 : Similaire à la Ligne 10, nous pouvons utiliser l'algorithme de Hopcroft-Karp [Hopcroft and Karp, 1973] pour trouver s'il existe un ensemble $\mathsf{LINKS}(\mathsf{succ}(x_0), \mathsf{succ}(z_0))$ dont chaque lien est vert. Donc, le temps d'exécution de l'Algorithme 3 par la Ligne 17 est $O(n^2 \times \sqrt{n})$.

□

Références

Carlos Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change : Partial meet contraction and revision functions. *Journal of symbolic Logic*, 50 :510–530, 1985.

A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. Dl-lite in the light of first-order logic. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 1*, AAAI'07, pages 361–366. AAAI Press, 2007.

F. Baader and W. Nutt. *Basic description logics*, pages 47–100. Cambridge University Press, 2003.

Franz Baader and Ulrike Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69(1) :5–40, 2001.

Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics as ontology languages for the semantic web. In *Festschrift in honor of Jörg Siekmann, Lecture Notes in Artificial Intelligence*, pages 228–248. Springer-Verlag, 2003.

Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook : Theory, Implementation, and Applications*. Cambridge University Press, 2007.

R. J. Brachrnan and II. J. Levesque. The tractability of subsumption in frame-based description languages. In *In Proceedings of the Fourth National Conference, on Artificial Intelligence*, pages 34–37, 1984.

Diego Calvanese, Giuseppe De Giacomo, Domenico Lemho, Maurizio Lenzerini, and Riccardo Rosati. Dl-lite : Tractable description logics for ontologies. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2*, AAAI'05, pages 602–607. AAAI Press, 2005.

Diego Calvanese, Giuseppe De Giacomo, Domenico Lemho, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. In *Proc. of KR*, page 260–270, 2006.

Diego Calvanese, Evgeny Kharlamov, Werner Nutt, and Dmitriy Zheleznyakov. Evolution of dl-lite knowledge bases. In *Proceedings of the 9th International Semantic Web Conference on The Semantic Web - Volume Part I*, ISWC'10, pages 112–128, Berlin, Heidelberg, 2010. Springer-Verlag.

M. Dalal. Investigations into a theory of knowledge base revision. *Proc. of AAAI*, pages 475–479, 1988.

Giuseppe De Giacomo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati. On the approximation of instance level update and erasure in description logics. In *Proceedings of AAAI*, pages 403–408, 2007.

Thomas Eiter and Georg Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence*, 57 :227–270, 1992.

Jérôme Euzenat. Revision in networks of ontologies. *Artif. Intell.*, 228 :195–216, 2015.

Philippe Fournier-Viger. Un modèle de représentation des connaissances à trois niveaux de sémantique pour les systèmes tutoriels intelligents. In *Mémoire de maîtrise (M.Sc.), Université de Sherbrooke, Sherbrooke, Canada*, 2005.

Jon William Freeman. *Improvements to Propositional Satisfiability Search Algorithms*. PhD thesis, University of Pennsylvania, 1995.

Birte Glimm, Ian Horrocks, and Boris Motik. Optimized description logic reasoning via core blocking. In *Proceedings of the 5th International Conference on Automated Reasoning*, IJCAR'10, pages 457–471, 2010.

Bernardo Cuenca Grau, Ernesto Jimenez-Ruiz, Evgeny Kharlamov, and Dmitry Zheleznyakov. Ontology Evolution Under Semantic Constraints. In *Proc. of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 137–147, 2012.

Volker Haarslev and Ralf Möller. *RACER System Description*, pages 701–705. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

Sven Ove Hansson. Kernel contraction. *J. Symbolic Logic*, 59(3) :845–859, 09 1994.

John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4) :225–231, 1973.

Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In *Proceedings of LPAR*. Springer, 1999.

Ian Horrocks, Ulrike Sattler, and Stephan Tobies. *Reasoning with Individuals for the Description Logic* SHIQ, pages 482–496. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

Ian R. Horrocks. Using an expressive description logic : Fact or fiction ? In *In Proc. of KR-98*, pages 636–647. Morgan Kaufmann, 1998.

Ian Horrocks. The description logic handbook. chapter Implementation and Optimization Techniques, pages 306–346. 2003.

A. K. Hudek and G. Weddell. Binary absorption in tableaux-based reasoning for description logics. In *In Proc. of the 2006 Description Logic Workshop (DL 2006*, 2006.

Hirofumi Katsuno and Alberto O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 53(3) :263–294, 1991.

Evgeny Kharlamov, Dmitriy Zheleznyakov, and Diego Calvanese. Capturing model-based ontology evolution at the instance level : The case of dl-lite. *J. Comput. Syst. Sci.*, 79(6) :835–872, September 2013.

Chan Le Duc, Myriam Lamolle, and Olivier Curé. A decision procedure for SHOIQ with transitive closure of roles. In *Proceedings of ISWC*, pages 264–279, 2013.

Marvin Minsky. A framework for representing knowledge. In John Haugeland, editor, *Mind Design : Philosophy, Psychology, Artificial Intelligence*, pages 95–128. MIT Press, Cambridge, MA, 1981.

Bernhard Nebel. Belief revision and default reasoning : Syntax-based approaches. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91). Cambridge, MA, USA, April 22-25, 1991.*, pages 417–428, 1991.

Guilin Qi and Jianfeng Du. Model-based revision operators for terminologies in description logics. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 891–897, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

Guilin Qi and Fangkai Yang. *A Survey of Revision Approaches in Description Logics*, pages 74–88. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

Guilin Qi, Weiru Liu, and David A. Bell. Knowledge base revision in description logics. In *European Conference on Logics in Artificial Inteligence*, pages 386–398, 2006.

Guilin Qi, Peter Haase, Zhisheng Huang, Qiu Ji, Jeff Z. Pan, and Johanna Volker. A kernel revision operator for terminologies — algorithms and evaluation. In *Proceedings of the 7th International Semantic Web Conference*, pages 419–434, 2008.

Guilin Qi, Zhe Wang, Kewen Wang, and Xuefeng Fu Zhiqiang Zhuang. Approximating model-based abox revision in dl-lite : Theory and practice. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 254–260. AAAI Press, 2015.

Marcio Moretto Ribeiro and Renata Wassermann. Base revision in description logics - preliminary results. *Proc. of IWOD*, pages 69–82, 2007.

Ken Satoh. Nonmonotonic reasoning by minimal belief revision. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 455–462, 1988.

Ulrike Sattler. A concept language extended with different kinds of transitive roles. pages 333–345. Springer-Verlag, 1996.

Andrea Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13 :141–176, 1994.

Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. In *Artificial Intelligence, 48*, pages 1–26, 1991.

Rob Shearer, Boris Motik, and Ian Horrocks. HermiT : A Highly-Efficient OWL Reasoner. In Alan Ruttenberg, Ulrile Sattler, and Cathy Dolbear, editors, *Proc. of the 5th*

Int. Workshop on OWL : Experiences and Directions (OWLED 2008 EU), Karlsruhe, Germany, October 26–27 2008.

Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet : A practical owl-dl reasoner. *Web Semantics : Science, Services and Agents on the World Wide Web*, 5(2) :51 – 53, 2007.

Stephan Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research*, 12 :199–217, 2000.

Stephan Tobies. Complexity results and practical algorithms for logics in knowledge representation. *PhD thesis, RWTH Aachen, Germany*, 2001.

Dmitry Tsarkov and Ian Horrocks. Efficient reasoning with range and domain constraints. In *In Proc. of the 2004 Description Logic Workshop (DL 2004*, pages 41–50, 2004.

Dmitry Tsarkov and Ian Horrocks. Fact++ description logic reasoner : System description. In *Proceedings of the Third International Joint Conference on Automated Reasoning*, IJCAR'06, pages 292–297, 2006.

Zhe Wang, Kewen Wang, and Rodney Topor. A new approach to knowledge base revision in dl-lite. *Proc. of 24th AAAI*, pages 369–374, 2010.

Zhe Wang, Kewen Wang, Zhiqiang Zhuang, and Guilin Qi. Instance-driven ontology evolution in dl-lite. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 1656–1662. AAAI Press, 2015.

Zhiqiang Zhuang, Zhe Wang, Kewen Wang, and Guilin Qi. Contraction and revision over dl-lite tboxes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1149–1155. AAAI Press, 2014.

Zhiqiang Zhuang, Zhe Wang, Kewen Wang, and Guilin Qi. Dl-lite contraction and revision. *Journal of Artificial Intelligence Research*, 56 :329–378, 2016.