Raisonnement fondé sur un tableau compressé pour les Logiques de Description

Chan Le Duc, Thinh Dong, Myriam Lamolle, Aurélien Bossard

Raisonnement fondé sur un tableau compressé pour les Logiques de Description

Chan Le Duc[1] **Thinh Dong**[1] **Myriam Lamolle**[1] **Aurélien Bossard**[1]

[1] LIASD - EA4383, IUT of Montreuil, University of Paris8, Saint Denis, France
{leduc, dong, lamolle, bossard}@iut.univ-paris8.fr

Résumé

Dans cet article, nous présentons un nouvel algorithme de raisonnement orienté but sur des ontologies d'expressivité $\mathcal{SHIQ}$ permettant les restrictions de cardinalité et les rôles inverses. Nous montrons qu'il s'exécute en ExpTime. En effet, dans le but de réduire la taille des modèles d'ontologie construits, notre algorithme utilise une structure, dite *tableau compressé*, composée de *star-types*, pour représenter intrinsèquement différents individus d'un modèle et ensuite compresser les individus équivalents dans un star-type. L'implémentation confirme que notre algorithme peut traiter des ontologies "difficiles" qui requièrent les raisonneurs existants de construire des modèles très larges. Nous proposons également une extension de l'algorithme à la logique $\mathcal{SHOIQ}$ permettant le traitement des nominaux.

Abstract

In this paper, we present a new goal-oriented algorithm for reasoning on ontologies expressed in the expressive description logic $\mathcal{SHIQ}$ allowing for number restrictions and inverse roles, and show that it runs optimally in ExpTime. In order to reduce the size of constructed models of an ontology, our algorithm uses a structure, namely *compressed tableau* composed of *star-types*, to represent intrinsically different individuals of a model and then compress similar individuals into a star-type. Our implementation shows that there are hard ontologies which would require existing reasoners to build a very large model for checking consistency, but can be processed by a reasoner based on our algorithm without cumbersome construction. We also discuss an extension of the algorithm to the logic $\mathcal{SHOIQ}$ allowing for nominals.

1 Introduction

Les formalismes fondés sur les logiques de description (DL) telles que OWL [9] sont largement utilisés pour représenter des ontologies intégrées dans des applications dites sémantiques. Une caractéristique intéressante des ontologies dites DL est de supporter des services d'inférence automatisés qui permettent aux concepteurs de détecter certaines erreurs et aux utilisateurs de découvrir de nouvelles connaissances. Plusieurs travaux ont été dédiés au problème du raisonnement pour des ontologies en logique $\mathcal{SHIQ}$ capables de gérer les opérateurs logiques de base, les rôles inverse, la transitivité, l'inclusion et les restrictions de cardinalité ("au-moins", "au-plus"). Tobies a montré que le problème de la cohérence pour une ontologie $\mathcal{SHIQ}$ est ExpTime-complet [13]. Horrocks *et al.* ont proposé une procédure de décision "tableau" pour $\mathcal{SHIQ}$ [6] qui a été implémentée dans des raisonneurs tels que FaCT++ [14], Pellet [12] et RACER [4]. Motik *et al.* ont développé de nouvelles techniques d'optimisation pour réduire les non déterminismes lors du raisonnement sur des ontologies volumineuses ayant des nominaux et des opérateurs sur les rôles [7]. Ces optimisations ont été implémentées dans le raisonneur HermiT [11], [3] et évaluées sur des ontologies très variées. Récemment, Nguyen a proposé un algorithme tableau ExpTime [8]. Sa méthode est fondée sur une *technique de cache global* qui utilise un graphe composé de noeuds OR et AND pour représenter les modèles. Pour réduire les non déterminismes liés aux restrictions de cardinalité, un système de contraintes d'inégalité a été conçu et une méthode de *programmation linéaire entière* a été utilisée pour vérifier si le système trouve une solution. A notre connaissance, cette méthode n'a pas encore été implémentée. De plus, l'extension de cet algorithme aux restrictions de cardinalité, au rôle inverse et aux nominaux n'est pas directe.

A notre connaissance, il n'existe pas de raisonneur fondé sur un algorithme *orienté but* ExpTime pour vérifier la cohérence d'une ontologie $\mathcal{SHIQ}$ avec des

nombres encodés en binaire. Rappelons qu'un algorithme est dit *orienté but* s'il construit des structures représentant un modèle ontologique de manière monotonique. Le principe essentiel des algorithmes tableau est la construction d'un graphe étiqueté avec la signature d'une ontologie représentant les modèles et une condition de blocage garantissant leur terminaison. Comme une telle condition de blocage est fondée sur la répétition des labels pour détecter les noeuds bloqués, la taille des graphes construits est bornée par une fonction doublement exponentielle en fonction de la taille de l'ontologie.

Dans cet article, nous présentons un algorithme dont la construction est fondée sur les notions de *star-types* et *frame* introduites par Pratt-Hartmann lors de la conception d'un algorithme NExpTime pour la logique $\mathcal{C}^2$ [10]. Au lieu de construire explicitement un graphe représentant un modèle comme les procédures de décision orientées but le font, notre algorithme construit une structure, dite *tableau-compressé* composée de *star-types*, chacun représentant un ensemble d'individus avec leurs voisins. Nous montrons que cet algorithme s'exécute en NExpTime malgré l'encodage binaire des nombres indiqués dans les restrictions "au-moins".

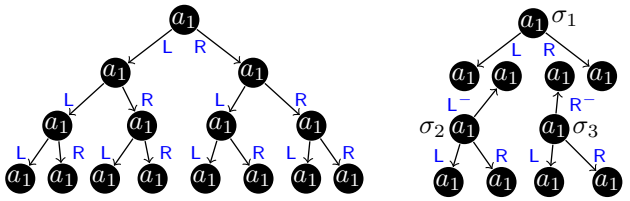

FIGURE 1 – Un arbre de complétion construit par un algorithme tableau en utilisant le blocage par paire (à gauche) ; star-types obtenus par compression (à droite)

Notre algorithme est la conséquence de deux intuitions à savoir : (i) *compression des individus similaires* ; deux individus d'un modèle généré par un algorithme tableau sont dits similaires s'ils ont le même label et leurs voisins sont appariés par paire. Les individus similaires peuvent être regroupés dans un star-type dont les rayons représentent les connexions possibles aux autres star-types. Puis, les relations de voisinage entre les star-types sont gérées par une *fonction d'appariement* qui permet à un star-type d'être connecté à d'autres via un rayon. Le nombre de star-types distincts construits ainsi est borné par une fonction doublement exponentielle en fonction de la taille de l'ontologie comme les nombres dans les restrictions "au-moins" sont codés en binaire. La Figure 1 montre comment compresser les noeuds aux deuxième et troisième niveaux de l'arbre dans deux star-types σ_1 et σ_2. Ces noeuds sont compressibles du fait qu'ils ont le même label (a_1) et les mêmes relations avec leurs voisins ; (ii) *retrait des star-types intégrables* ; un star-type σ est dit *intégrable* dans un ensemble de star-types Σ si chaque rayon de σ appartient à l'ensemble des rayons d'un star-type de Σ. Dans ce cas, il n'est pas nécessaire d'ajouter σ dans Σ pour représenter un modèle. Il est donc suffisant de stocker dans un tableau-compressé un nombre exponentiel de star-types distincts puisque chaque star-type d'un tel tableau doit contenir un rayon qui n'est pas inclus dans un autre star-type de ce tableau. La figure 2 montre la capacité d'intégration d'un star-type σ_1 dans $\{\sigma_2, \sigma_3\}$ par décomposition de σ_1. Cette décomposition est possible car deux rayons ρ_1 (contenant un rôle L) et ρ_2 (contenant un rôle R) de σ_1 sont inclus dans σ_2 et σ_3. Ceci permet de remplacer chaque connexion à partir d'un star-type ω vers σ_1 via ρ_1 ou ρ_2 par une nouvelle connexion de ω à σ_2 ou σ_3.

Cet article est organisé comme suit. La section 2 introduit une définition formelle $\mathcal{SHIQ}$ et les notations nécessaires pour définir les structures sur lesquelles est fondé notre algorithme. Dans la section 3, nous étendons certaines structures présentées par Pratt-Hartmann [10] pour $\mathcal{C}^2$ à $\mathcal{SHIQ}$ et nous fournissons une définition des tableaux compressés pour $\mathcal{SHIQ}$. La section 4 décrit l'algorithme pour la vérification de la cohérence d'une ontologie $\mathcal{SHIQ}$ et montre qu'il s'exécute en ExpTime. Nous expérimentons aussi notre algorithme sur plusieurs ontologies qui peuvent requérir des algorithmes standard basés tableau construisant un modèle doublement exponentiel. La section 5 présente une implémentation de l'algorithme ainsi que quelques tests. Enfin, nous présentons une synthèse concernant les résultats obtenus ainsi que nos futurs travaux.

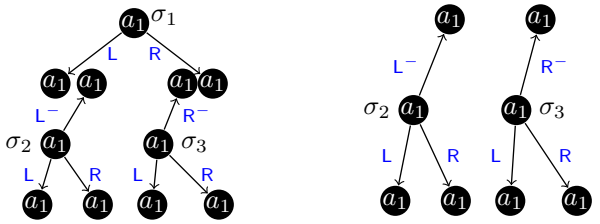

FIGURE 2 – Star-type σ_1 intégrable à l'ensemble des star-types $\{\sigma_2, \sigma_3\}$

2 Préliminaires

Cette section présente la syntaxe et la sémantique de $\mathcal{SHIQ}$. Soit $\mathbf{R}$ un ensemble non vide de *noms de rôles* et $\mathbf{R}_+ \subseteq \mathbf{R}$ un ensemble de noms de rôles transitifs. Nous utilisons $\mathbf{R}_\mathsf{I} = \{R^- \mid R \in \mathbf{R}\}$ pour

noter un ensemble de rôles inverses. Chaque élément de $\mathbf{R} \cup \mathbf{R}_\mathsf{I}$ est dit un $\mathcal{SHIQ}$-rôle. Pour simplifier les notations de rôles inverses imbriqués, nous définissons une fonction $\mathsf{Inv}(S) = R^-$ si $S = R$; et $\mathsf{Inv}(S) = R$ si $S = R^-$ où $R \in \mathbf{R}$. Un *axiome d'inclusion de rôle* est de la forme $R \sqsubseteq S$ pour deux $\mathcal{SHIQ}$-rôles R et S (potentiellement inverses). Une *hiérarchie de rôles* $\mathcal{R}$ est un ensemble fini d'axiomes d'inclusion de rôles. Une relation sous-rôle $\underline{\circledast}$ 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 renvoie true 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 dit *simple* selon $\mathcal{R}$ si $\mathsf{Trans}(R) = \mathsf{false}$. Une interprétation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consiste en un ensemble non vide $\Delta^\mathcal{I}$ (*domaine*) et une fonction $\cdot^\mathcal{I}$ qui associe chaque 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 $\mathcal{R}$, notée $\mathcal{I} \models \mathcal{R}$, 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 concepts*. L'ensemble de $\mathcal{SHIQ}$-concepts est défini par récurrence 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 des $\mathcal{SHIQ}$-concepts, R est un $\mathcal{SHIQ}$-rôle et S est un rôle simple par rapport à une hiérarchie de rôles. Nous notons $\bot$ 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 sous-ensemble de $\Delta^\mathcal{I}$ telle que $\top^\mathcal{I} = \Delta^\mathcal{I}$, $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$, $(C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}$, $(\neg C)^\mathcal{I} = \Delta^\mathcal{I} \backslash C^\mathcal{I}$, $(\exists R.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \exists y \in \Delta^\mathcal{I}, \langle x, y\rangle \in R^\mathcal{I} \wedge y \in C^\mathcal{I}\}$, $(\forall R.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \forall y \in \Delta^\mathcal{I}, \langle x, y\rangle \in R^\mathcal{I} \Rightarrow y \in C^\mathcal{I}\}$, $(\geq n S.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid |\{y \in C^\mathcal{I} \mid \langle x, y\rangle \in S^\mathcal{I}\}| \geq n\}$, $(\leq n S.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid |\{y \in C^\mathcal{I} \mid \langle x, y\rangle \in S^\mathcal{I}\}| \leq n\}$ où $|S|$ représente la cardinalité d'un ensemble S.

Un axiome $C \sqsubseteq D$ est appelé une inclusion générale de concept(GCI) où C, D sont $\mathcal{SHIQ}$-concepts (éventuellement complexes), et un ensemble fini de GCIs est appelé une terminologie $\mathcal{T}$. Une interprétation $\mathcal{I}$ satisfait une GCI $C \sqsubseteq D$, notée $\mathcal{I} \models (C \sqsubseteq D)$, si $C^\mathcal{I} \subseteq D^\mathcal{I}$. Une interprétation $\mathcal{I}$ est un modèle de $\mathcal{T}$, notée $\mathcal{I} \models \mathcal{T}$, si $\mathcal{I}$ satisfait chaque GCI dans $\mathcal{T}$. Une paire $(\mathcal{T}, \mathcal{R})$ est dite une *ontologie* $\mathcal{SHIQ}$, notée $\mathcal{O} = (\mathcal{T}, \mathcal{R})$, si $\mathcal{R}$ est une hiérarchie de rôles $\mathcal{SHIQ}$ et $\mathcal{T}$ est une terminologie $\mathcal{SHIQ}$. Une ontologie $\mathcal{O} = (\mathcal{T}, \mathcal{R})$ est dite cohérente s'il existe un modèle $\mathcal{I}$ à la fois de $\mathcal{T}$ et de $\mathcal{R}$, *i.e.*, $\mathcal{I} \models \mathcal{T}$ et $\mathcal{I} \models \mathcal{R}$. Nous utilisons $\mathbf{R}(\mathcal{T}, \mathcal{R})$ pour noter l'ensemble de tous les noms de rôle apparaissant dans $\mathcal{T}, \mathcal{R}$ avec leur inverse.

Lors de la construction des structures pour représenter un modèle d'une ontologie, il est nécessaire d'ajouter de nouveaux concepts qui ne sont pas présents initialement dans l'ontologie. Les ensembles suivants encapsulent de tels concepts. Tous les concepts sont en *forme normale négative* (NNF), *i.e.*, la négation apparaît uniquement au début des noms de concept. Tout $\mathcal{SHIQ}$-concept peut être transformé en son équivalent NNF en utilisant les lois de DeMorgan, les dualités entre les restrictions existentielle et universelle et les dualités entre les restrictions "au-moins" et "au-plus". $\mathsf{nnf}(C)$ peut être calculé en un temps polynomial en fonction de la taille de C [1]. Pour chaque concept C, nous notons la nnf de C par $\mathsf{nnf}(C)$ et la nnf de $\neg C$ par $\dot\neg C$. Soit D un $\mathcal{SHIQ}$-concept en NNF et $(\mathcal{T}, \mathcal{R})$ une ontologie. La fermeture $\mathbf{CL}(D)$ est définie comme le plus petit ensemble qui contient tous les sous-concepts de D incluant D. L'ensemble $\mathbf{CL}(D, \mathcal{R})$ est l'union des ensembles suivants : $\mathbf{CL}(D)$, $\{\dot\neg C \mid C \in \mathbf{CL}(D)\}$, $\{\forall S.C \mid \forall R.C \in \mathbf{CL}(D), S \underline{\circledast} R\}$, $\{\forall S.C \mid \dot\neg\forall R.C \in \mathbf{CL}(D), S \underline{\circledast} R\}$ où S apparaît dans $\mathcal{T}$ ou $\mathcal{R}$. L'ensemble $\mathbf{CL}(\mathcal{T}, \mathcal{R})$ est l'union de tous les $\mathbf{CL}(\mathsf{nnf}(\neg C \sqcup D), \mathcal{R})$ pour chaque $C \sqsubseteq D \in \mathcal{T}$. Pour chaque restriction de concept "au-moins" $\geq nS.C$ apparaissant dans $\mathbf{CL}(\mathsf{nnf}(\neg C \sqcup D), \mathcal{R})$, un ensemble de nouveaux noms de concepts $\{C^i_{(\geq nS.C)} \mid 0 \leq i \leq \lceil \log\, n+1 \rceil\}$ ainsi que leurs compléments sont ajoutés à $\mathbf{CL}(\mathsf{nnf}(\neg C \sqcup D), \mathcal{R})$. Nous utilisons $\mathscr{C}^I_{(\geq nS.C)}$ pour noter un sous-ensemble de l'union $\{C^i_{(\geq nS.C)} \mid 0 \leq i \leq \lceil \log\, n+1 \rceil\} \cup \{\neg C^i_{(\geq nS.C)} \mid 0 \leq i \leq \lceil \log\, n+1 \rceil\}$ pour $I \subseteq \{0, \cdots, \lceil \log\, n+1 \rceil\}$ tel que $C^i_{(\geq nS.C)} \in \mathscr{C}^I_{(\geq nS.C)}$ ssi $\neg C^i_{(\geq nS.C)} \notin \mathscr{C}^I_{(\geq nS.C)}$. Par construction, la cardinalité de $\mathbf{CL}(\mathcal{T}, \mathcal{R})$ reste polynomialement bornée par la taille de l'ontologie même si les entiers dans les restrictions "au-moins" sont codés en binaire.

3 Tableau-compressé pour $\mathcal{SHIQ}$

Cette section décrit une structure appelée *tableau-compressé* composée de *star-types* qui sera utilisée pour représenter un modèle d'une ontologie $\mathcal{SHIQ}$. La construction de cette structure pour $\mathcal{SHIQ}$ est adaptée des notions de *frame* et *star-type* introduites par Pratt-Hartmann [10] pour le fragment deux-variable de la logique du premier ordre des quantificateurs de cardinalité $\mathcal{C}^2$. La définition suivante décrit un star-type générique dépendant seulement de la signature d'une ontologie $\mathcal{SHIQ}$.

Définition 1 (star-type)**.** *Soit* $(\mathcal{T}, \mathcal{R})$ *une ontologie* $\mathcal{SHIQ}$*, un* star-type *est une paire* $\sigma = \langle\lambda(\sigma), \xi(\sigma)\rangle$*, où* $\lambda(\sigma) \in 2^{\mathbf{CL}(\mathcal{T},\mathcal{R})}$ *est appelé* core label*,* $\xi(\sigma) = (\langle r_1, l_1\rangle, \cdots, \langle r_d, l_d\rangle)$ *est un d-tuple avec* $(r_1, l_1) \in 2^{\mathbf{R}(\mathcal{T},\mathcal{R})} \times 2^{\mathbf{CL}(\mathcal{T},\mathcal{R})}$ *pour* $1 \leq i \leq d$*. Une paire* $\rho =$

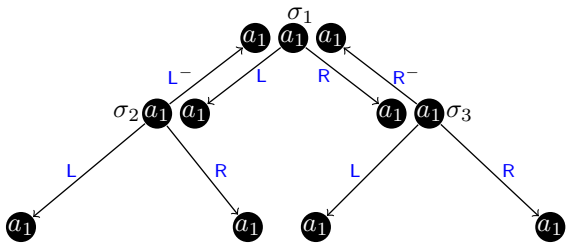

FIGURE 3 – σ_1 et σ_2 sont appariés via 2 rayons contenant les rôles L et L^- ; σ_1 et σ_3 sont appariés via 2 rayons contenant les rôles R et R^-.

$\langle r(\rho), l(\rho)\rangle$ est un rayon *de σ si $\langle r(\rho), l(\rho)\rangle = \langle r_i, l_i\rangle$ pour $1 \leq i \leq d$.*

- *Un star-type σ est* chromatique *si $\rho \neq \rho'$ implique $l(\rho) \neq l(\rho')$ pour deux rayons ρ, ρ' de σ.*
- *Deux star-types σ, σ' sont* équivalents *si $\lambda(\sigma) = \lambda(\sigma')$, et s'il y a une bijection π entre $\xi(\sigma)$ et $\xi(\sigma')$ telle que $\pi(\rho) = \rho'$ implique $r(\rho') = r(\rho)$ et $l(\rho') = l(\rho)$.* ◁

Si un star-type σ est chromatique, $\xi(\sigma)$ peut être considéré comme un ensemble de rayons du fait que leurs rayons sont distincts et non ordonnés. Nous pouvons voir un star-type σ comme l'ensemble des individus x satisfaisant tous les concepts dans $\lambda(\sigma)$, et chaque rayon ρ de σ correspond à un individu "voisin" x_i de x tel que $r(\rho)$ soit le label du lien entre x et x_i ; et x_i satisfait tous les concepts dans $l(\rho)$. Dans ce cas, nous disons que x *satisfait* σ. Dans la figure 3, les star-types σ_1 et σ_2 sont voisins. Il est possible qu'un star-type σ ait deux voisins star-types différents σ' et σ'' via un rayon ρ. Dans ce cas, il y a au moins deux individus satisfaisant σ qui se connectent à deux individus satisfaisant σ' et σ'' respectivement.

Définition 2 (star-type valide)**.** *Soit $(\mathcal{T}, \mathcal{R})$ une ontologie $\mathcal{SHIQ}$. Soit σ un star-type pour $(\mathcal{T}, \mathcal{R})$ où $\sigma = \langle \lambda(\sigma), \xi(\sigma)\rangle$. Le star-type σ est* valide *si σ est chromatique et les conditions suivantes sont satisfaites.*

1. *Si $C \sqsubseteq D \in \mathcal{T}$ alors $\mathsf{nnf}(\neg C \sqcup D) \in \lambda(\sigma)$.*
2. *$\{A, \neg A\} \not\subseteq \lambda$ pour chaque nom de concept A où $\lambda = \lambda(\sigma)$ ou $\lambda = l(\rho)$ pour chaque $\rho \in \xi(\sigma)$.*
3. *Si $C_1 \sqcap C_2 \in \lambda(\sigma)$ alors $\{C_1, C_2\} \subseteq \lambda(\sigma)$.*
4. *Si $C_1 \sqcup C_2 \in \lambda(\sigma)$ alors $\{C_1, C_2\} \cap \lambda(\sigma) \neq \emptyset$.*
5. *Si $\exists R.C \in \lambda(\sigma)$ alors il y a un rayon $\rho \in \xi(\sigma)$ tel que $C \in l(\rho)$ et $R \in r(\rho)$.*
6. *Pour chaque rayon $\rho \in \xi(\sigma)$, si $R \in r(\rho)$ et $R \sqsubseteq S$ alors $S \in r(\rho)$.*
7. *Si $\forall R.C \in \lambda(\sigma)$ et $R \in r(\rho)$ pour un rayon $\rho \in \xi(\sigma)$ alors $C \in l(\rho)$.*
8. *Si $\forall R.D \in \lambda(\sigma)$, $S \sqsubseteq R$, $\mathsf{Trans}(S)$ et $R \in r(\rho)$ pour un rayon $\rho \in \xi(\sigma)$ alors $\forall S.D \in l(\rho)$.*
9. *Si $(\geq nS.C) \in \lambda(\sigma)$ alors il y a n rayons distincts $\rho_1, \cdots, \rho_n \in \xi(\sigma)$ tels que $\{C\} \cup \mathscr{C}^{I_i}_{(\geq nS.C)} \subseteq l(\rho_i)$, $S \in r(\rho_i)$ pour tout $1 \leq i \leq n$; et $I_j, I_k \subseteq \{0, \cdots, \lceil log\ n + 1\rceil\}$, $I_j \neq I_k$ pour tout $1 \leq j < k \leq n$.*
10. *Si $(\leq nS.C) \in \lambda(\sigma)$ alors il n'existe pas $n+1$ rayons $\rho_0, \cdots, \rho_n \in \xi(\sigma)$ tel que $C \in l(\rho_i)$ et $S \in r(\rho_i)$ pour tout $0 \leq i \leq n$.*
11. *Si $(\leq nS.C) \in \lambda(\sigma)$ et il y a un rayon $\rho \in \xi(\sigma)$ tel que $S \in r(\rho)$ alors $C \in l(\rho)$ ou $\doteq C \in l(\rho)$.* ◁

Si nous utilisons un star-type σ pour représenter un ensemble d'individus X d'un modèle ontologique, la validité de σ implique la satisfaction de tous les concepts dans $\lambda(\sigma)$, *i.e.* chaque $x \in X$ satisfait *sémantiquement* tous les concepts dans $\lambda(\sigma)$. En fait, chaque condition dans Définition 2 représente les sémantiques d'un constructeur dans $\mathcal{SHIQ}$. Notons que la condition 9 assure la propriété chromatique des star-types valides du fait que les concepts $\mathscr{C}^I_{(\geq nS.C)}$ avec $I \subseteq \{0, \cdots, \lceil log\ n+1\rceil\}$ sont utilisés pour assigner un label distinct de concept à chaque rayon.

Définition 3 (Tableau-compressé)**.** *Soit $(\mathcal{T}, \mathcal{R})$ une ontologie $\mathcal{SHIQ}$. Un* tableau-compressé *pour $(\mathcal{T}, \mathcal{R})$ est une paire $\mathsf{T} = \langle \mathcal{N}, \Omega\rangle$, où*

1. *$\mathcal{N}$ est un ensemble de star-types valides tel que σ ne soit pas équivalent à σ' pour tout $\sigma, \sigma' \in \mathcal{N}$;*
2. *pour chaque star-type $\sigma \in \mathcal{N}$, il y a un rayon $\rho \in \xi(\sigma)$ tel que $\rho \notin \xi(\sigma')$ pour tout $\sigma' \in \mathcal{N} \setminus \{\sigma\}$ avec $\lambda(\sigma') = \lambda(\sigma)$;*
3. *Ω est une fonction appelée* appariement *qui associe chaque paire (σ, ρ) avec $\sigma \in \mathcal{N}$, $\rho \in \xi(\sigma)$ à un ensemble de paires $\langle(\sigma_1, \rho_1), \cdots, (\sigma_m, \rho_m)\rangle$ avec $\sigma_i \in \mathcal{N}$, $\rho_i \in \xi(\sigma_i)$ tel que $l(\rho) = \lambda(\sigma_i)$, $l(\rho_i) = \lambda(\sigma)$ et $r^-(\rho) = r(\rho_i)$ pour tout $1 \leq i \leq m$ où $r^-(\rho) = \{\mathsf{Inv}(R) \mid R \in r(\rho)\}$. Dans ce cas, nous disons que ρ est apparié à ρ_i.* ◁

Selon la condition 1 de la Définition 3, tous les star-types d'un tableau-compressé sont distincts. La condition 2 assure que $|\mathcal{N}|$ est borné par le nombre de triplets distincts (λ, r, λ') avec $\lambda, \lambda' \in 2^{\mathbf{CL}(\mathcal{T},\mathcal{R})}$ et $r \in 2^{\mathbf{R}(\mathcal{T},\mathcal{R})}$. Il y a une relation entre un tableau-compressé et un arbre de complétion construit par un algorithme tableau. En effet, un tableau-compressé peut être obtenu d'un arbre de complétion en compressant les noeuds dont les voisins sont appariés par paire. Inversement, une fonction d'appariement comme décrite par la condition 3 fournit une possibilité d'extension d'un tableau-compressé à un arbre de complétion. Cette fonction d'appariement peut être considérée comme une généralisation de la condition de blocage de différents algorithmes tableau tel que le blocage d'ancêtre [5], le blocage "n'importe où" [7] car

l'arc connectant un noeud bloquant à son prédécesseur est apparié à un arc connectant le noeud bloqué à son prédécesseur. De plus, nous n'avons pas besoin de stocker dans un tableau-compressé un star-type σ qui est intégrable dans d'autres star-types σ', σ'' (*i.e.* $\xi(\sigma) \subseteq \xi(\sigma') \cup \xi(\sigma'')$) comme décrit en section 2 du fait que chaque $\Omega(\sigma, \rho)$ peut être remplacé par $\Omega(\sigma', \rho)$ ou $\Omega(\sigma'', \rho)$. Dans ce cas, si σ est enlevé du tableau, toutes les conditions de la Définition 3 restent vraies.

Lemme 1. *Soit $(\mathcal{T}, \mathcal{R})$ une ontologie $\mathcal{SHIQ}$. $(\mathcal{T}, \mathcal{R})$ est cohérente ssi $(\mathcal{T}, \mathcal{R})$ a un tableau-compressé.*

Résumé de preuve. *Supposons qu'il y ait un tableau-compressé $\mathsf{T} = \langle \mathcal{N}, \Omega \rangle$ for $(\mathcal{T}, \mathcal{R})$. Grâce à la fonction d'appariement Ω, nous pouvons passer par les star-types pour construire un modèle en forme d'arbre par "démêlage". Inversement, supposons qu'il y ait un modèle $(\Delta, .^{\mathcal{I}})$ pour $(\mathcal{T}, \mathcal{R})$. Pour chaque $s \in \Delta$, nous définissons un star-type $\sigma(s)$ valide comme suit :*

- *Si $s \in C^{\mathcal{I}}$ alors $C \in \lambda(\sigma(s))$ avec $C \in 2^{\mathbf{CL}(\mathcal{T},\mathcal{R})}$. Pour chaque concept $(\geq n_i R_i.C_i) \in \lambda(\sigma)$, nous notons $S_{(\geq n_i R_i.C_i)}(s) = \{t^1_{n_i}, \cdots, t^{n_i}_{n_i}\} \subseteq \Delta$ tel que $\langle s, t^j_{n_i} \rangle \in R_i^{\mathcal{I}}$ et $\{C_i, \mathscr{C}^{I_j}_{(\geq n_i R_i.C_i)}\} \subseteq \lambda(\sigma(t^j_{n_i}))$ avec $I_j \subseteq \{0, \cdots, \lceil log\ n_i + 1 \rceil\}$ pour $1 \leq j \leq n_i$, et $I_j \neq I_{j'}$ pour $1 \leq j < j' \leq n_i$.*
- *Pour chaque $t \in \bigcup_{(\geq nR.C) \in \mathcal{L}(s)} S_{(\geq nR.C)}(s)$, nous ajoutons un rayon $\rho = \langle r, l \rangle$ à $\xi(\sigma(s))$ et $\rho' = \langle r^-, l' \rangle$ à $\xi(\sigma(t))$ tel que $l = \lambda(\sigma(t))$, $l' = \lambda(\sigma(s))$ et $r = \{S \mid (s,t) \in S^{\mathcal{I}} \wedge R \sqsubseteq S\}$. Dans ce cas, nous ajoutons $(\rho', \sigma(t))$ à $\Omega(\rho, \sigma(s))$ et $(\rho, \sigma(s))$ à $\Omega(\rho', \sigma(t))$.*
- *Si $\sigma(s) \equiv \sigma(s')$ alors σ' peut être enlevé et Ω reste une fonction d'appariement. De plus, si $\xi(\sigma) \subseteq \xi(\sigma_1) \cup \cdots \cup \xi(\sigma_m)$ alors σ peut être enlevé et Ω reste aussi une fonction d'appariement.* □

4 Construction de tableaux-compressés

La section précédente a fourni les propriétés d'un tableau-compressé qui nous permet de caractériser un modèle d'une ontologie $\mathcal{SHIQ}$. Nous présentons ici un algorithme orienté but pour la vérification de la cohérence d'une ontologie $\mathcal{SHIQ}$ en essayant de construire un tableau-compressé.

Avant de décrire l'algorithme en détail, nous définissons les structures de données suivantes :
SAT est un ensemble de star-types valides ; UNSAT est un ensemble de star-types invalides (ou non saturés), *i.e.* toute les conditions de la Définition 1 ne sont pas satisfaites. Donc, $\mathsf{SAT} \cap \mathsf{UNSAT} = \emptyset$. Pour chaque $\sigma \in \mathsf{SAT} \cup \mathsf{UNSAT}$, nous définissons un ensemble de triplets $\mathsf{TR}(\sigma) = \{(l', r, l) \mid \lambda = \lambda(\sigma), r = r(\rho), l = l(\rho), \rho \in \xi(\sigma)\}$. TRIPLE est l'union de tous les $\mathsf{TR}(\sigma)$ pour $\sigma \in \mathsf{SAT} \cup \mathsf{UNSAT}$. MATCHED est un ensemble de triplets (l'_1, r_1, l_1) tels que $(l'_1, r_1, l_1) \in$ MATCHED si et seulement si $(l'_1, r_1, l_1) \in \mathsf{TR}(\sigma)$ avec $\sigma \in$ SAT et il existe un autre triplet $(l'_2, r_2, l_2) \in$ MATCHED avec $l'_1 = l_2$, $l'_2 = l_1$ et $r_1 = r_2^-$. Pour deux star-types σ, σ' avec $\tau \in \mathsf{TR}(\sigma)$, $\tau' \in \mathsf{TR}(\sigma')$, nous pouvons former un duo $\langle (\sigma, \tau), (\sigma', \tau') \rangle$. Si τ et τ' sont appariés selon la Définition 3, il est appelé un *duo apparié*, un *duo non apparié* autrement. UNMATCHED est un ensemble de duos non appariés $\langle (\sigma, \tau), (\sigma', \tau') \rangle$ tel que $\sigma, \sigma' \in$ SAT $\cup$ UNSAT. PROCESSED est un sous-ensemble de UNMATCHED. Tous ces ensembles sont initialisés à vide. Ils seront modifiés de façon monotonique par les procédures suivantes.

Algorithm 1: Vérification de la cohérence d'une ontologie

Input : Une ontologie $\mathcal{SHIQ}$ $(\mathcal{T}, \mathcal{R})$
Output: Cohérence de $(\mathcal{T}, \mathcal{R})$

```
1 Créer un duo ⟨(σ1, ρ1), (σ2, ρ2)⟩ et appeler Paving-duo pour le paver ;
2 foreach ⟨(σ, ρ), (σ', ρ')⟩ ∈ UNMATCHED \ PROCESSED do
3     Appeler Paving-duo pour paver ⟨(σ, ρ), (σ', ρ')⟩;
4     if Detecting-compressed-tableau trouve un tableau-compressé then
5         return YES;
6 return NO;
```

- `Saturating-star-type` Pour chaque propriété de la Définition 2, nous définissons une règle pour saturer un star-type. Ces règles agissent comme des règles d'expansion d'algorithme tableau [6]. Par exemple, pour chaque star-type σ si $\exists R.C \in \lambda(\sigma)$ et il n'y a pas de triplet $(l', r, l) \in \mathsf{TR}(\sigma)$ tel que $R \in r$ et $C \in l'$ alors nous ajoutons un rayon (r, l) à $\xi(\sigma)$ avec $R \in r$ et $C \in l$. Notons que la propriété chromatique d'un star-type peut remplacer la relation d'inégalité utilisée dans les algorithmes tableau pour éviter de fusionner des voisins créés par la $\geq$-règle. En appliquant les règles non déterministes correspondant aux propriétés 4, 10 and 11 dans la Définition 2, les star-types non saturés peuvent être ajoutés à UNSAT. Par exemple, en appliquant la règle correspondante à la propriété 10, deux rayons peuvent être choisis pour être fusionnés. Les autres choix de rayons pour une fusion peuvent entraîner la création de nouveaux star-types ajoutés dans UNSAT et mis à jour dans UNMATCHED.
- `Paving-duo` Cette procédure commence en choisissant une paire (τ, σ) à partir de UNMATCHED \ PROCESSED. Puis, elle crée un nouveau star-type σ' ayant seulement un triplet $\tau' = (l, r^-, l')$

si $\tau = (l', r, l)$, et un duo $\langle(\tau, \sigma), (\tau', \sigma')\rangle$. Donc, τ et τ' sont appariés et le duo est pavé via τ et τ'. Cette procédure appelle `Saturating-star-type` pour saturer les star-types σ et σ' tout en préservant l'appariement du duo. Si aucune paire de star-types n'est apparié aux autres (τ, σ) de UNMATCHED \ PROCESSED. Sinon, elle retourne un duo apparié $\langle(\tau_1, \sigma_1), (\tau_2, \sigma_2)\rangle$ de $\langle(\tau, \sigma), (\tau', \sigma')\rangle$. Dans ce cas, il ajoute τ_1, τ_2 à MATCHED et met à jour SAT, UNSAT et UNMATCHED. Enfin, il ajoute (τ, σ) à PROCESSED.

– `Embedding-star-type` Cette procédure est appelée quand un nouveau star-type σ doit être ajouté à SAT ou UNSAT. Elle vérifie si la condition 2 de la Définition 3 est satisfaite. Si la condition n'est pas respectée, σ n'est pas ajouté dans SAT ou UNSAT.

– `Detecting-compressed-tableau` Pour chaque mise à jour de SAT, la procédure vérifie si un tableau-compressé est trouvé dans cet ensemble. Ceci permet de terminer l'algorithme sans considérer tous les cas de non déterminisme. Une implémentation naïve travaille ainsi. La procédure commence en créant une copie SAT′ de SAT, en traversant chaque duo de UNMATCHED pour trouver les triplets τ non appariés et enlevant de SAT′ tous les star-types σ contenant τ. Ensuite, il enlève de SAT′ tous les star-types σ' contenant un triplet qui appartient à un star-type précédemment retiré, et ainsi de suite. Un tableau-compressé est détecté si SAT′ reste non vide après retrait des star-types par cette procédure. Par construction, chaque triplet $\tau \in \mathsf{TR}(\sigma)$ est apparié à un autre triplet $\tau' \in \mathsf{TR}(\sigma')$ avec $\sigma, \sigma' \in \mathsf{SAT}'$. Ceci implique qu'une fonction d'appariement comme décrite dans la condition 3 de la Définition 3 peut être définie à partir des star-types restants dans SAT′. Nous avons implémenté une version optimisée de cette procédure qui indexe les star-types obtenus lors de vérifications précédentes.

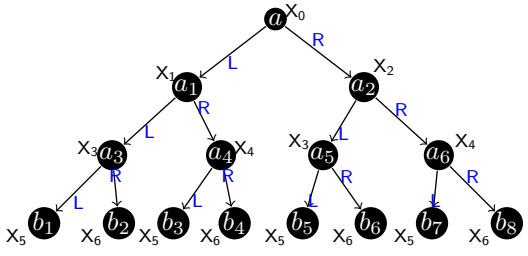

FIGURE 4 – Arbre de complétion pour l'ontologie FAT-2

L'algorithme 1 est le programme principal qui appelle les procédures décrites ci-dessus. Pour illustrer les star-types générés par l'algorithme et comment ils sont appariés pour former un tableau-compressé, nous considérons les ontologies, appelées FAT-n, constituées ds axiomes (2)-(6). L'assertion (1) indique seulement que l'algorithme commence par la création d'un star-type avec le concept $(B_0 \sqcap \cdots \sqcap B_{k-1} \sqcap C)$ dans son core. Un arbre de complétion $\mathsf{T}_{\mathsf{FAT\text{-}k}}$ de profondeur 2^k (*i.e.*, ayant 2^{2^k} feuilles) est construit par un algorithme tableau [6] pour FAT-k. Chaque noeud à un niveau $i < 2^k$ de $\mathsf{T}_{\mathsf{FAT\text{-}k}}$ a un label correspondant à la valeur binaire de i si nous utilisons les concepts $B_0, \cdots, B_{k-1}, \neg B_0, \cdots, \neg B_{k-1}$ pour représenter des chiffres binaires : B_i pour 0 et $\neg B_i$ pour 1. Par exemple, si $i = 3$ et $k = 4$ alors le label du noeud au niveau i contient $B_3, B_2, \neg B_1, \neg B_0$. Les axiomes (1), (3) et (4) propage C à chaque noeud de $\mathsf{T}_{\mathsf{FAT\text{-}k}}$. En particulier, l'axiome (2) impose que chaque noeud contienne exclusivement B_i ou $\neg B_i$ pour $0 \leq i \leq k-1$. Les axiomes (3) et (4) permettent que l'algorithme génère à partir de chaque noeud un L-successeur et un R-successeur. Ces deux successeurs ne sont pas "fusionnable" du fait que l'un contient D et l'autre contient $\neg D$. Les axiomes (5) et (6) assurent que s'il y a un noeud courant contenant les concepts $X_0, \cdots, X_{k-1}$ avec $X_i \in \{B_i, \neg B_i\}$ qui représente une valeur binaire $\overline{b_{k-1} \cdots b_0}$ alors les deux successeurs doivent contenir les concepts $Y_0, \cdots, Y_{k-1}$ avec $Y_i \in \{B_i, \neg B_i\}$ qui représente la valeur binaire $\overline{b_{k-1} \cdots b_0} + 1$. Plus précisément, les axiomes (5) garantissent que si le bit du poids faible est modifié de 0 en 1 alors les autres bits restent inchangés. Quant aux les axiomes (6), ils garantissent que si tous les i bits du poids faible sont modifiés de 1 en 0 et le bit de rang $(i+1)$ de 0 en 1 alors les autres bits doivent rester inchangés. Ces propriétés forcent l'incrémentation des nombres binaires correspondant aux labels des noeuds de la racine jusqu'aux feuilles. La Figure 5 présente $\mathsf{T}_{\mathsf{FAT\text{-}2}}$. Notons que, dans le label des noeuds, nous ne mentionnons pas les concepts $\neg C \sqcup D$ ajoutés pour chaque axiome $C \sqsubseteq D$. Pour la rendre plus lisible, nous définissons les labels de noeud comme suit.

$\mathsf{Y}_1 = \exists \mathsf{L}.(\neg \mathsf{B}_0 \sqcap \mathsf{C} \sqcap \mathsf{D})$, $\mathsf{Y}_2 = \exists \mathsf{L}.(\neg \mathsf{B}_0 \sqcap \mathsf{C} \sqcap \neg \mathsf{D})$,
$\mathsf{Z}_1 = \exists \mathsf{L}.(\mathsf{B}_0 \sqcap \neg \mathsf{B}_1 \sqcap \mathsf{C} \sqcap \mathsf{D})$,
$\mathsf{Z}_2 = \exists \mathsf{L}.(\mathsf{B}_0 \sqcap \neg \mathsf{B}_1 \sqcap \mathsf{C} \sqcap \neg \mathsf{D})$,
$\mathsf{X}_0 = \{\mathsf{B}_0, \mathsf{B}_1, \mathsf{C}, \mathsf{Y}_1, \mathsf{Y}_2, \forall \mathsf{L}.\mathsf{B}_1, \forall \mathsf{R}.\mathsf{B}_1\}$,
$\mathsf{X}_1 = \{\neg \mathsf{B}_0, \mathsf{B}_1, \mathsf{C}, \mathsf{D}, \mathsf{Z}_1, \mathsf{Z}_2\}$,
$\mathsf{X}_2 = \{\neg \mathsf{B}_0, \mathsf{B}_1, \mathsf{C}, \neg \mathsf{D}, \mathsf{Z}_1, \mathsf{Z}_2\}$,
$\mathsf{X}_3 = \{\mathsf{B}_0, \neg \mathsf{B}_1, \mathsf{C}, \mathsf{D}, \mathsf{Y}_1, \mathsf{Y}_2, \forall \mathsf{L}.(\neg \mathsf{B}_1), \forall \mathsf{R}.(\neg \mathsf{B}_1)\}$,
$\mathsf{X}_4 = \{\mathsf{B}_0, \neg \mathsf{B}_1, \mathsf{C}, \neg \mathsf{D}, \mathsf{Y}_1, \mathsf{Y}_2, \forall \mathsf{L}.(\neg \mathsf{B}_1), \forall \mathsf{R}.(\neg \mathsf{B}_1)\}$,
$\mathsf{X}_5 = \{\neg \mathsf{B}_0, \neg \mathsf{B}_1, \mathsf{C}, \mathsf{D}\}$, $\mathsf{X}_6 = \{\neg \mathsf{B}_0, \neg \mathsf{B}_1, \mathsf{C}, \neg \mathsf{D}\}$.

$$(B_0 \sqcap \cdots \sqcap B_{k-1} \sqcap C)(a) \tag{1}$$

$$C \sqsubseteq \prod_{0\leq i\leq k-1} (B_i \sqcup \neg B_i) \quad (2)$$

$$B_0 \sqcap C \sqsubseteq \exists L.(\neg B_0 \sqcap C \sqcap D) \sqcap \exists R.(\neg B_0 \sqcap C \sqcap \neg D) \quad (3)$$

$$\neg B_0 \sqcap \cdots \sqcap \neg B_{i-1} \sqcap B_i \sqcap C \sqsubseteq \exists L.(B_0 \sqcap \cdots \sqcap B_{i-1} \sqcap \neg B_i \sqcap C \sqcap D) \sqcap \exists R.(B_0 \sqcap \cdots \sqcap B_{i-1} \sqcap \neg B_i \sqcap C \sqcap \neg D) \quad \text{pour } 1 \leq i \leq k-1 \quad (4)$$

$$B_0 \sqcap X_i \sqcap \exists S.\neg B_0 \sqcap \exists S.\neg X_i \sqsubseteq \bot \text{ pour } S \in \{L, R\}, X_i \in \{B_i, \neg B_i\} \text{ for } 0 < i \leq k-1 \quad (5)$$

$$\prod_{0\leq l\leq i-1} (\neg B_l \sqcap \exists S.B_l) \sqcap B_i \sqcap \exists S.\neg B_i \sqcap X_j \sqcap \exists S.\neg X_j \sqsubseteq \bot \quad X_j \in \{B_j, \neg B_j\}, S \in \{L, R\},\ 0 < i < j \leq k-1 \quad (6)$$

Notons qu'il y a $(k-1)$ axiomes définis par (4), $2(k-1)$ axiomes définis par(5) et $4\Sigma_{i=1}^{k-1} i(k-i)$ axiomes définis par (6). Ceci implique que $\mathsf{FAT\text{-}k}$ est constituée de $3 + 3(k-1) + 4\Sigma_{i=1}^{k-1} i(k-i)$ axiomes.

Si nous exécutons l'Algorithme 1 pour $\mathsf{FAT\text{-}k}$, il y a au moins 4 star-types générés pour chaque niveau de $\mathsf{T}_{\mathsf{FAT\text{-}k}}$. Dès lors, nous pouvons obtenir un tableau-compressé qui est composé d'au moins 4×2^k de star-types. La Figure 5 montre un tableau-compressé pour $\mathsf{T}_{\mathsf{FAT\text{-}2}}$. Il est constitué de 11 star-types $\sigma_i (0 \leq i \leq 10)$. Les lignes en pointillé représentent l'appariement entre deux star-types via deux rayons. Nous observons que le star-type σ_8 (σ_9, σ_4, ou σ_5) représente les noeuds b_1 et b_5 (b_2 et b_6, b_3 et b_7, ou b_4 et b_8, respectivement) de l'arbre de la Figure 5.

Prouvons maintenant que l'Algorithme 1 est une procédure de décision pour vérifier la cohérence d'une ontologie $\mathcal{SHIQ}$.

Arrêt Dans la description des procédures, nous avons prouvé qu'elles se terminent. Soient $\ell = |(\mathcal{T}, \mathcal{R})|$, $n = |2^{\mathbf{CL}(\mathcal{T},\mathcal{R})}|$ et $m = |2^{\mathbf{R}(\mathcal{T},\mathcal{R})}|$. Ceci implique que $n, m \leq \mathcal{O}(2^\ell)$. Comme un triplet est composé de deux labels de concept et un label de rôle, $|\mathsf{TRIPLE}| \leq n^2 m \leq \mathcal{O}(2^\ell)$. Si les nombres dans les restrictions "au-moins" sont codés en binaire alors $|\mathsf{SAT}|, |\mathsf{UNSAT}| \leq \mathcal{O}(2^{2^\ell})$. Cependant, la procédure `Embedding-star-type` assure que chaque star-type $\sigma \in \mathsf{SAT} \cup \mathsf{UNSAT}$ doit contenir un triplet qui n'est pas inclus dans les autres star-types. Ainsi, $|\mathsf{SAT}|, |\mathsf{UNSAT}| \leq \mathcal{O}(|\mathsf{TRIPLE}|) \leq \mathcal{O}(2^\ell)$. De plus, nous obtenons $|\mathsf{UNMATCHED}| \leq (|\mathsf{SAT}| + |\mathsf{UNSAT}|)|\mathsf{TRIPLE}| \leq \mathcal{O}(2^\ell)$ par construction. Chaque itération dans la boucle (ligne 2) de l'Algorithme 1 déplace un élément de $\mathsf{UNMATCHED}$ vers $\mathsf{PROCESSED}$. Par conséquent, le nombre d'itérations de la boucle est borné par $\mathcal{O}(2^\ell)$.

Correction Si l'Algorithme 1 peut construire un tableau-compressé $\mathsf{T} = \langle \mathcal{N}, \Omega \rangle$ pour une ontologie $(\mathcal{T}, \mathcal{R})$ alors elle est cohérente. C'est une conséquence directe du Lemme 1.

Complétude Si une ontologie $(\mathcal{T}, \mathcal{R})$ est cohérente alors l'Algorithme 1 peut construire un tableau-compressé $\mathsf{T} = \langle \mathcal{N}, \Omega \rangle$. Avant d'argumenter, nous voudrions mettre en exergue une subtilité relative à la gestion non-déterministe dans la preuve. Quand l'algorithme traite un non-déterminisme (*e.g.* $\sqcup$-rule), il peut faire un "mauvais" choix alors qu'un meilleur choix doit exister dans le tableau-compressé. Dans ce cas, nous allons montrer que l'algorithme peut toujours revenir en arrière pour faire un "bon" choix.

Selon le Lemme 1, $(\mathcal{T}, \mathcal{R})$ a un tableau-compressé $\mathsf{T}' = \langle \mathcal{N}', \Omega' \rangle$. Nous utilisons une fonction π pour maintenir les relations entre les star-types de T' et ceux dans SAT et de $\mathsf{UNMATCHED} \setminus \mathsf{PROCESSED}$ construits par l'algorithme. Grâce à cette fonction, nous pouvons obtenir un tableau-compressé quand l'algorithme se termine. L'algorithme (avec `Paving-duo`) commence par construire un duo valide (un tel duo peut toujours être initialisé à partir d'un star-type et d'un rayon) noté $\langle(\omega_1, \nu_1), (\omega_2, \nu_2)\rangle$. L'algorithme met dans $\mathsf{MATCHED}$ deux triplets ρ_1, ρ_2, dans SAT deux star-types σ_1, σ_2, dans UNSAT les star-types générés par les règles non-déterministes et dans $\mathsf{UNMATCHED}$ les duos créés à partir des star-types $\mathsf{SAT} \cup \mathsf{UNSAT}$ via les triplets restants (qui sont différents de ρ_1, ρ_2). Nous définissons $\pi(\sigma_1) = \sigma'_1$ et $\pi(\sigma_2) = \sigma'_2$ tel que $\sigma'_1, \sigma'_2 \in \mathcal{N}'$ soient appariés dans T' (de tels star-types existent toujours dans T'), et σ_i est inclus dans σ'_i pour $1 \leq i \leq 2$, noté $\sigma_i \prec \sigma'_i$, ce qui signifie que $\lambda(\sigma_i) \subseteq \lambda(\sigma'_i)$ et qu'il y a une injection δ de $\mathsf{TR}(\sigma_i)$ à $\mathsf{TR}(\sigma'_i)$ telle que, pour chaque $\rho \in \mathsf{TR}(\sigma_i)$, nous avons $r(\rho) \subseteq r(\delta(\rho))$ et $l(\rho) \subseteq l(\delta(\rho))$.

Supposons que l'algorithme extrait de $\mathsf{UNMATCHED} \setminus \mathsf{PROCESSED}$ un duo $\langle(\omega_1, \nu_1), (\omega_2, \nu_2)\rangle$ tel que $\pi(\omega_1) = \omega'_1$ et $\pi(\omega_2) = \omega'_2$ soient respectés (nous devons montrer que cette propriété est respectée après chaque mise-à-jour de π, *i.e.* π il met en correspondance dans T' seulement les star-types dans les duos appartenant à $\mathsf{UNMATCHED} \setminus \mathsf{PROCESSED}$). La procédure `Paving-duo` commence par paver ce duo en appelant `Saturating-star-type` afin de satisfaire ω_1 et ω_2 en préservant leurs appariements. Cette procédure génère de $\langle(\omega_1, \nu_1), (\omega_2, \nu_2)\rangle$ un nouveau duo pavé noté $\langle(\sigma_1, \rho_1), (\sigma_2, \rho_2)\rangle$. Quand une règle déterministe est appliquée à ω_i, $\sigma_i \prec \omega'_i$ est respecté. Dans ce cas, nous changeons $\pi(\omega_i) = \omega'_i$ en $\pi(\sigma_i) = \omega'_i$ $(1 \leq i \leq 2)$. Supposons qu'une règle non-déterministe est appliquée à ω_i. Nous considérons alors les deux possibilités suivantes : (i) l'algorithme fait un "mauvais" choix tel que $\sigma_i \prec \omega'_i$ ne soit plus respecté. Selon la description de la procédure `Saturating-star-type`,

un nouveau star-type σ'_i est ajouté à UNSAT tel que $\sigma'_i \prec \omega'_i$ soit respecté. Dans ce cas, nous changeons $\pi(\omega_i) = \omega'_i$ en $\pi(\sigma'_i) = \omega'_i$; (ii) l'algorithme fait un "bon" choix tel que $\sigma_i \prec \omega'_i$ est respecté. Dans ce cas, nous changeons $\pi(\omega_i) = \omega'_i$ en $\pi(\sigma_i) = \omega'_i$, et σ_i est ajouté à SAT. Comme ci-dessus, l'algorithme met à jour MATCHED, SAT (voire UNSAT), UNMATCHED, et ajoute $\langle(\omega_1, \nu_1), (\omega_2, \nu_2)\rangle$ à PROCESSED. Par monocité de la saturation, le duo $\langle(\sigma_1, \rho_1), (\sigma_2, \rho_2)\rangle$ qui a été ajouté à UNMATCHED n'appartient pas à PROCESSED. Ceci assure que la fonction π met en correspondance seulement les star-types dans SAT et de UNMATCHED \ PROCESSED à T′.

L'algorithme s'arrête quand UNMATCHED \ PROCESSED $= \emptyset$; soit quand chaque triplet $\rho \in \mathsf{TR}(\sigma)$ de chaque $\sigma \in$ SAT est apparié à un autre triplet $\rho' \in \mathsf{TR}(\sigma')$ avec $\sigma' \in$ SAT (autrement, UNMATCHED \ PROCESSED $\neq \emptyset$). Par conséquent, l'algorithme a construit un tableau-compressé du fait que T′ est un tableau-compressé.

Théorème 1. *Soit $(\mathcal{T}, \mathcal{R})$ une ontologie $\mathcal{SHIQ}$. L'Algorithme 1 vérifie la cohérence de $(\mathcal{T}, \mathcal{R})$ et s'exécute en* ExpTime *en fonction de la taille de l'ontologie $(\mathcal{T}, \mathcal{R})$.*

Résumé de preuve. *Nous avons prouvé que l'Algorithme 1 vérifie la cohérence de $(\mathcal{T}, \mathcal{R})$. Pour prouver qu'il s'exécute en* ExpTime *de la taille de $(\mathcal{T}, \mathcal{R})$, nous avons besoin de montrer que le nombre d'itérations de la boucle de l'Algorithme 1 est borné par $\mathcal{O}(2^\ell)$ avec $\ell = |(\mathcal{T}, \mathcal{R})|$ et chaque procédure* `Saturating-star-type`, `Paving-duo`, `Detecting-compressed-tableau` *s'exécutent en* ExpTime. *En effet, le nombre d'itérations de la boucle de l'Algorithme 1 est borné par $\mathcal{O}(2^\ell)$ si* |UNMATCHED| $\leq \mathcal{O}(2^\ell)$ *et* PROCESSED *augmente après chaque itération. La procédure* `Saturating-star-type` *s'exécute en* ExpTime *puisque $|\lambda(\sigma)| \leq \mathcal{O}(\ell)$;* `Paving-duo` *s'exécute en* ExpTime *puisque* `Saturating-star-type` *s'exécute en* ExpTime *(puisque le cardinal du label de core de chaque startype est borné par ℓ),* |UNMATCHED| $\leq \mathcal{O}(2^\ell)$ *et aucun duo* UNMATCHED \ PROCESSED *n'est choisi deux fois. Enfin,* `Detecting-compressed-tableau` *s'exécute en* ExpTime *puisque* |SAT| $\leq \mathcal{O}(2^\ell)$. □

5 Expérimentations

Nous avons implémenté un prototype, appelé Staré, fondé sur l'Algorithme 1. Dans sa version actuelle, il fournit uniquement un service de vérification de cohérence d'ontologie. Nous avons utilisé les raisonneurs HermiT-1.3.8 et Pellet-2.3.3 comme implémentations

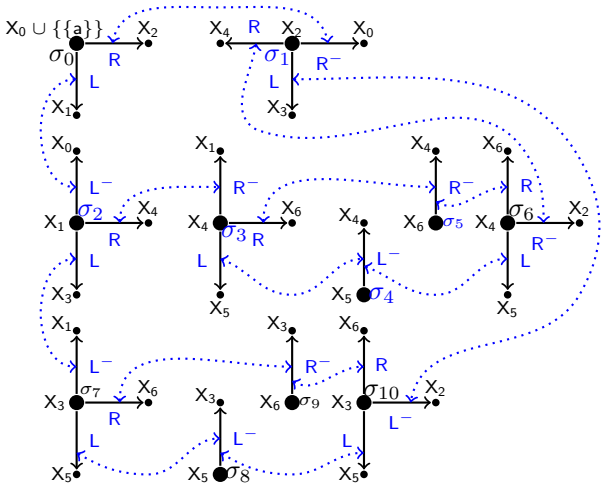

FIGURE 5 – Tableau-compressé pour l'ontologie FAT-2

de référence en Java des algorithmes tableau ou hypertableau pour $\mathcal{SHIQ}$.

Nous avons créé des ontologies "artificielles" comme décrites en Section 4, nommées `FAT-n` avec `n` $\in \{5, 7, 10\}$ pour tester Staré. De plus, nous avons effectué des expériences pour tester sa consommation de mémoire avec des ontologies existantes telles que `GALEN-doctored` dérivée de `full-galen` [1].

La Table 1 donne les caractéristiques des ontologies utilisées pour les tests et la Table 2 les résultats obtenus. Tous les tests ont été effectué sur un ordinateur DELL avec 8 processeurs Intel 3.4GHz et 32Gb RAM sous Ubuntu.

TABLE 1 – Caractéristiques des ontologies

Ontologie	Caractéristiques		
	Concepts	Roles	Axioms
FAT-5	9	2	48
FAT-7	11	2	94
FAT-10	14	2	193
Galen-doctored	2748	413	3937

L'objectif de Staré est de montrer que l'Algorithme 1 fonctionne sur des ontologies "difficiles" qui pourraient requérir des raisonneurs existants comme HermiT [11], [3] et Pellet [12] de construire de très grands modèles. Les résultats obtenus montrent que Staré devient meilleur que les autres raisonneurs pour les ontologies FAT-8, FAT-10 bien que Staré consomme beaucoup plus de mémoire. Cependant, il est moins bon que les autres raisonneurs lors de calcul sur des ontologies impliquant de nombreux non déterminismes. Nous pensons que la consommation mémoire de Staré est causée par le stockage des structures des star-types

1. En effet, les raisonneurs ne savent pas traiter les individus.

TABLE 2 – Résultats des tests (nous écrivons - si le raisonneur met plus de 15 minutes)

Ontologie	Temps de vérification (en sec.)		
	Staré	HermiT	Pellet
FAT-5	4	2	2
FAT-7	47	130	118
FAT-8	202	1245	1530
FAT-10	3101	-	-
Galen-doctored	3	1	2

et des triplets qui ne sont pas dans les modèles. *A contrario* des raisonneurs fondés sur les algorithmes tableau qui stockent seulement les structures nécessaires et redémarrent la construction des noeuds d'un tableau quand un clash apparaît, Staré rejette juste les star-types contenant un clash et stocke ceux qui ont mené au clash. Staré consomme beaucoup de mémoire pour stocker les strustures qu'il peut potentiellement réutiliser plus tard.

Nous avons aussi testé Staré sur des ontologies volumineuses telles que `full-galen` qui a donné des résultats moins intéressants. En effet, Staré doit traiter dans ce cas 1951 axiomes généraux qui peuvent lui exiger de construire beaucoup de star-types avant de trouver un tableau-compressé. Ces axiomes doivent être absorbés par une technique d'absorption implémentée dans HermiT et Pellet. Ce mauvais comportement de Staré relatif à la gestion du non-déterminisme pourrait s'expliquer par l'absence de techniques avancées comme l'absorption binaire, la normalisation d'axiomes dans sa version actuelle.

6 Conclusion et discussion

Nous avons présenté un algorithme de vérification de la cohérence d'une ontologie $\mathcal{SHIQ}$ et montré qu'il s'exécute en ExpTime en fonction de la taille de l'ontologie. La conception de cet algorithme est fondé sur les deux idées principales suivantes : (i) la compression d'individus similaires pour former un star-type et le retrait de ceux qui peuvent être contenus dans un ensemble de star-types ; (ii) la seconde qui est cruciale est la réduction d'un nombre doublement exponentiel de star-types distincts d'un tableau-compressé en un nombre exponentiel de star-types distincts. Nous avons implémenté notre algorithme dans Staré et effectué des test sur des ontologies pour lesquelles les raisonneurs existants peuvent générer de larges modèles. Les résultats expérimentaux obtenus confirment que notre algorithme peut éviter une construction de modèles doublement exponentiels en fonction de la taille des entrées. Cependant, la question ouverte restante est comment la représentation compacte des modèles résultant de l'algorithme se répercute sur une réduction éventuelle des non-déterminismes ?

Une implémentation d'autres services d'inférence comme l'implication logique, la classification est en cours. Nous croyons que de tels services fondés sur notre algorithme peut gérer des ontologies "difficiles" si chaque vérification pour une implication logique $C \sqsubseteq D$ ne devrait pas entraîner la construction de beaucoup plus de star-types. Nous pensons que plus de structures seront mises en cache au bon moment plus facilement elles seront réutilisées pour construire différents tableau-compressés lors de la classification d'une ontologie.

La logique $\mathcal{SHOIQ}$ est une extension de $\mathcal{SHIQ}$ en y ajoutant des nominaux, *i.e.* qu'elle permet d'exprimer un concept comme une liste d'individus $\{o_1, \cdots, o_k\}$. Par conséquent, un algorithme pour raisonner en $\mathcal{SHOIQ}$ doit gérer les concepts tels que$\exists R.\{o\}$, $\forall R.\{o\}$ qui devraient propager les nominaux sur des structures représentant les modèles. Si chaque nominal doit être unique dans un modèle, *i.e.* $|\{o\}^{\mathcal{I}}| = 1$ pour une interprétation$\mathcal{I}$, la propagation des nominaux peut amener à fusionner des structures qui représentent le même nominal. De plus, l'unicité des nominaux peut entraîner des conditions de blocage plus difficiles à satisfaire du fait que les parties duplicables d'une structure pour représenter un modèle ne doivent pas contenir de nominaux.

En ayant ceci à l'esprit, nous pouvons outrepasser ces difficultés (i) en fusionnant les star-types qui contiennent le même nominal ; et (ii) en détectant un sous-ensemble de star-types valides, appelés *cercle*, tel que chaque star-type d'un cercle contient aucun nominal et s'apparie avec un autre star-type du même cercle via chaque rayon. Dans ce cas, nous pouvons concevoir un modèle à partir d'un tel cercle si chaque star-type d'un cercle peut être dupliqué à l'infini. Donc, une ontologie $\mathcal{SHOIQ}$ est cohérente si un cercle est détecté dans un ensemble de star-types valides. Nous prévoyons d'étendre Staré pour traiter $\mathcal{SHOIQ}$ et fournir d'autres services d'inférence tels l'implication logique et que la classification.

Une autre extension de Staré vise à fournir un nouveau service d'inférence qui calcule des modèles *représentatifs* d'une ontologie. Par exemple, Dong *et al.* [2] ont proposé un algorithme pour la révision d'ontologie qui requiert la construction de différents modèles au lieu d'un seul. Un tel service d'inférence essaie de retrouver la cohérence d'une ontologie en la révisant par adjonction de nouveaux axiomes incompatibles avec les axiomes déjà existants. Ceci requiert un algorithme de révision pour calculer un ensemble de modèles représentatifs caractérisant les sémantiques d'un ontologie. Un tel service sera très utile pour une approche qui

nécessite d'approximer sémantiquement une ontologie par une autre.

Références

[1] Franz Baader and Werner Nutt. Basic description logics. In *The Description Logic Handbook : Theory, Implementation and Applications (2nd edition)*, pages 47–104. Cambridge University Press, 2007.

[2] Thinh Dong, Chan Le Duc, Philippe Bonnot, and Myriam Lamolle. Tableau-based revision over $\mathcal{SHIQ}$ tboxes. In *Logic for Programming, Artificial Intelligence, and Reasoning - 20th International Conference, LPAR-20 2015, Suva, Fiji, November 24-28, 2015, Proceedings*, pages 575–590, 2015.

[3] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. Hermit : An owl 2 reasoner. *J. of Automated Reasoning*, 53(3) :245–269, 2014.

[4] Volkers Haarslev and Ralf Moller. Racer system description. In *Proc. of the Int. Joint on Automated Reasoning (IJCAR 2001)*, pages 701–705. Springer, 2001.

[5] Ian Horrocks and Ulrike Sattler. A tableau decision procedure for $\mathcal{SHOIQ}$. *Journal Of Automated Reasoning*, 39(3) :249–276, 2007.

[6] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In *Proceedings of the International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 1999)*. Springer, 1999.

[7] Boris Motik, Rob Shearer, and Ian Horrocks. Hypertableau reasoning for description logics. *J. of Artificial Intelligence Research*, 36 :165–228, 2009.

[8] Linh Anh Nguyen. A tableau method with optimal complexity for deciding the description logic SHIQ. In *Advanced Computational Methods for Knowledge Engineering*, pages 331–342, 2013.

[9] Peter Patel-Schneider, P. Hayes, and Ian Horrocks. Owl web ontology language semantics and abstract syntax. In *W3C Recommendation*, 2004.

[10] Ian Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Language and Information*, 14(3) :369–395, 2005.

[11] 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.

[12] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet : a pratical OWL-DL reasoner. *Journal of Web Semantics*, 5(2) :51–53, 2007.

[13] Stephan Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research*, 12 :199–217, 2000.

[14] Dmitry Tsarkov and Ian Horrocks. FaCT++ description logic reasoner : System description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer, 2006.