Découverte des meilleures couvertures d'un concept en utilisant une terminologie. Application à la découverte de services web sémantiques.

Christophe Rey

Université Blaise Pascal - Clermont II

Ecole Doctorale
Sciences Pour l'Ingénieur de Clermont-Ferrand

THÈSE

présentée par

Christophe Rey

Formation doctorale :
Informatique, productique, imagerie médicale

pour obtenir le grade de

Docteur d'Université

Spécialité : Informatique

Découverte des meilleures couvertures d'un concept en utilisant une terminologie Application à la découverte de services web sémantiques

Soutenue publiquement le mardi 14 décembre 2004 devant le jury :

Rapporteurs :	Mme Marie-Christine Rousset
	M. Bernd Amann
Examinateurs :	M. Mohand-Saïd Hacid (président du jury)
	M. Michel Schneider (directeur de thèse)
	M. Farouk Toumani (co-directeur de thèse)
	M. Alain Léger (co-directeur de thèse)
Invité :	M. Jean-Marc Petit

Mis en page avec la classe thloria.

Remerciements

Tout d'abord, je tiens à remercier très chaleureusement Farouk Toumani pour son écoute, sa grande patience et son exigence scientifique constante. Les nombreuses heures passées à discuter, à construire, à démontrer, bref à avancer ensemble, resteront pour moi des moments inoubliables. Pleins d'excitation intellectuelle, mais aussi de doutes et de remises en question profondes, ils auront toujours été enrichissants. Ils sont ma formation, et j'espère m'en montrer digne dans l'avenir.

Je remercie aussi Michel Schneider pour sa disponibilité, sa gentillesse et pour m'avoir permis de mener cette thèse dans les meilleures conditions qui soient, ainsi qu'Alain Léger pour la confiance qu'il m'a accordée. Je salue sincèrement Mohand-Saïd Hacid pour m'avoir donné de son temps et de ses conseils dans une période mouvementée.

Je pense également à Patrick Keller, à mes collègues de bureau (actuels et passés), et aux enseignants et chercheurs de l'IUT GTR de Clermont, en particulier Michel Misson et Antonio Freitas, pour leur convivialité et leur sympathie.

Enfin, je tiens à exprimer ma reconnaissance envers toutes les personnes qui, de près ou de loin, m'ont permis de bénéficier d'une allocation de recherche sans laquelle ce travail n'aurait pu voir le jour.

A Marie, pour son amour et son soutien, même dans les moments difficiles,
et à Mathilde, pour son sourire, et pour avoir fait ses nuits rapidement...

Liste des tableaux

Table des figures

Table des matières

Introduction

Contexte général

Les *services web sémantiques* [62, 1, 35] constituent la nouvelle génération des technologies du web pour l'intégration d'applications inter-entreprises au fil de l'eau. Ils se situent à la convergence de deux domaines de recherche importants : les services web et le web sémantique.

La notion de *service web* désigne une application mise à disposition par un fournisseur et invocable sur internet par des clients (utilisateurs ou autres services). L'ambition portée par les services web est de permettre une plus grande interopérabilité entre applications sur internet. On envisage ainsi des services web capables, automatiquement, de se découvrir et d'être découverts, de négocier entre eux ou de se composer en des services plus complexes. Ce sont les problématiques de la découverte, de l'exécution et de la composition dynamique de services. Actuellement, les services web sont mis en œuvre au travers de trois technologies standard : WSDL[1], UDDI[2] et SOAP[3]. Ces technologies facilitent la description, la découverte et la communication entre services. Cependant, cette infrastructure de base ne permet pas encore aux services web de tenir leur promesse d'une gestion largement automatisée. Cette automatisation est pourtant essentielle pour faire face aux exigences du passage à l'échelle, d'une forte réactivité dans un environnement hautement dynamique et de la volonté de réduire les coûts de développement et de maintenance des services. Fondamentalement, pour franchir cette nouvelle étape dans l'automatisation, il manque aux services web un moyen d'être décrits d'une manière compréhensible par une machine.

Le *web sémantique* [17] est une vision du web dans laquelle toute information possède une sémantique. Cette sémantique a été pensée pour être compréhensible par une machine. Il en résulte qu'en raisonnant logiquement sur cette sémantique, la machine peut manipuler les informations (par exemple les rechercher ou les classifier) d'une manière plus pertinente qu'avec les techniques syntaxiques classiques. On peut, par exemple, détecter des incohérences ou faire le lien entre des concepts synonymes sur la base de leurs descriptions. Pour cela, il est nécessaire d'utiliser des langages qui possèdent une sémantique formelle. De plus, la construction d'ontologies est préconisée pour résoudre les problèmes d'hétérogénéité entre concepts. Ainsi, les technologies existantes du web sémantique consistent-elles essentiellement en des langages comme RDF[4] et OWL[5] pour la description de ressources et d'ontologies sur le web.

Appliqués aux services web, les principes du web sémantique doivent donc permettre de décrire la sémantique de leurs fonctionnalités et de leur comportement. Les raisonnements induits constituent une proposition pour automatiser les différentes tâches de leur cycle de vie. Ceci

[1]"Web Services Description Language". Voir http ://www.w3.org/TR/wsdl/.
[2]"Universal Description, Discovery and Integration". Voir http ://www.uddi.org/.
[3]"Simple Object Access Protocol". Voir http ://www.w3.org/TR/soap12-part0/.
[4]" Resource Description Framework". Voir http ://www.w3.org/RDF/.
[5]"Web Ontology Language". Voir http ://www.w3.org/TR/owl-features/

constitue la problématique des services web sémantiques [62, 26, 1, 35, 18].

Meilleures couvertures d'un concept en utilisant une terminologie

Dans cette thèse, on s'intéresse au problème de la découverte dynamique de services web sémantiques [43, 66, 75, 18, 48, 58]. A partir des descriptions d'une requête d'un utilisateur et de services disponibles, le problème est de chercher les services les plus pertinents par rapport à la requête. Nous formalisons ce problème comme un nouveau raisonnement dans le contexte des logiques de description [64]. Les logiques de description fournissent un cadre reconnu pour la représentation de la sémantique des informations du web et pour les raisonnements [3, 36, 45]. Elles constituent un formalisme central du web sémantique. Elles sont par exemple à la base du langage OWL récemment normalisé par le W3C.

Le raisonnement que nous proposons est une nouvelle instance du cadre général de la réécriture de concepts en utilisant une terminologie [9]. On l'a appelé la *découverte des meilleures couvertures d'un concept en utilisant une terminologie*. Il s'énonce de la manière suivante : "étant donnés un concept Q (une requête) et une terminologie $\mathcal{T}$ de concepts (les services), comment récrire Q en toutes les conjonctions de concepts de $\mathcal{T}$ (les ensembles de services) qui *sémantiquement se rapprochent le plus* de Q ?". La notion de proximité sémantique entre concepts est formellement définie en s'appuyant sur l'opérateur de différence sémantique entre concepts des logiques de description [74]. Elle consiste à minimiser les différences entre Q et ses réécritures potentielles. Cela permet de maximiser le nombre d'informations communes et donc d'assurer la plus grande proximité possible entre Q et ses réécritures. Pour une requête Q, on obtient donc tous les sous-ensembles de services de $\mathcal{T}$ qui contiennent le plus possible d'informations communes avec Q et le moins possible d'informations spécifiques par rapport à Q.

D'autres réécritures pour les logiques de description sont définies dans [9, 12, 72, 39, 42]. Elles sont utilisées notamment dans le contexte des systèmes de médiation, pour résoudre le problème de la réécriture de requête en utilisant des vues. Elles utilisent les relations de subsomption ou d'équivalence pour définir les réécritures. Par rapport à ces dernières, l'intérêt de notre approche réside dans l'utilisation d'un critère plus général, car plus souple, que la subsomption ou l'équivalence. En effet, en cas d'existence de réécritures équivalentes ou subsumées, ces dernières sont trouvées préférentiellement puisqu'elles maximisent par nature l'information commune entre la requête et ses réécritures. En cas d'absence de réécriture équivalente ou subsumée, seul un concept Q n'ayant aucune information en commun avec ceux de la terminologie n'aboutira à aucun résultat. Ainsi, la notion de meilleures couvertures permet un processus de réécriture plus flexible qui généralise les approches existantes. La découverte dynamique de services qui en découle est donc une découverte flexible. Cette flexibilité, préconisée dans [66], est particulièrement utile dans le contexte versatile du web.

Résultats de la thèse

Les principaux résultats de cette thèse sont les suivants.

- Nous étudions d'abord le problème de la découverte des meilleures couvertures en utilisant une terminologie dans le cas des logiques de description ayant la propriété de subsomption structurelle. Cette propriété permet de caractériser une famille de langages pour lesquels l'opération de différence entre concepts est sémantiquement unique[6] et facilement calcu-

[6] L'opération de différence est sémantiquement unique quand toute différence entre deux concepts n'aboutit qu'à un seul concept résultat, modulo l'équivalence.

lable par une différence ensembliste. Nous formalisons notre problème dans ce contexte. Nous montrons que c'est un problème NP-Difficile et qu'il peut se ramener au problème de la recherche des transversaux minimaux de coût minimal dans un hypergraphe où les sommets possèdent un coût. Pour le résoudre, nous proposons un algorithme appelé *computeBCov*.

- Nous étudions ensuite le problème de la découverte des meilleures couvertures dans le cas de la logique de description $\mathcal{ALN}$. Cette logique est connue pour offrir un bon compromis entre son expressivité et la complexité de ses raisonnements. Cependant, la possibilité d'exprimer le concept inconsistant de manière non triviale dans $\mathcal{ALN}$ rend le problème plus complexe que précédemment. La première raison est que l'opération de différence entre concepts n'est plus sémantiquement unique. Il faut donc gérer la multiplicité des résultats de la différence (par exemple pouvoir comparer des ensembles de descriptions). La seconde raison est que certaines réécritures peuvent être inconsistantes. Il faut donc éviter de les générer. Nous commençons par donner un moyen pour calculer la différence sémantique dans $\mathcal{ALN}$. A partir de ce résultat, nous formalisons la découverte des meilleures couvertures dans le contexte d'$\mathcal{ALN}$. Nous montrons que le problème ainsi défini est NP-Difficile. Pour le résoudre, nous proposons un algorithme nommé *computeALNBCov*. Cet algorithme consiste en une réduction progressive de l'espace de recherche des meilleures couvertures, basée sur le calcul des cas d'inconsistances entre concepts de la terminologie.

L'exploitation pratique de ces résultats a été entreprise à travers un prototype implémentant *computeBCov*. Ce prototype a notamment été expérimenté au sein du projet européen MKBEEM[7]. Le but de MKBEEM est l'étude et la réalisation d'une plate-forme de commerce électronique, basée sur des techniques de traitement de la langue naturelle et de représentation de connaissances et raisonnement. Les services de médiation offerts dans MKBEEM sont centrés sur les domaines de la vente par correspondance et du tourisme. Ce dernier a fourni des scenarii qui ont permis la validation de la découverte de services par les meilleures couvertures. Quantitativement, des expérimentations menées sur des ontologies générées aléatoirement ont montré que *computeBCov* limitait l'explosion combinatoire inhérente au problème, condition sine qua non du passage à l'échelle. En dehors de MKBEEM, *computeBCov* a aussi été expérimenté dans un contexte pair-à-pair pour la découverte de e-catalogs et leur interrogation flexible.

Organisation du mémoire

Le chapitre 1 est consacré à la présentation des logiques de description, leur syntaxe, leur sémantique et les raisonnements associés.

Le chapitre 2 présente et motive informellement la découverte des meilleures couvertures d'un concept en utilisant une terminologie. Une solution à ce problème pour les logiques de description ayant la propriété de subsomption structurelle est donnée au chapitre 3. Une solution pour le langage $\mathcal{ALN}$ est donnée au chapitre 4. Le chapitre 5 positionne alors les résultats précédents par rapport à ceux des autres travaux sur la réécriture.

Le chapitre 6 présente l'application des meilleures couvertures à la découverte de services web sémantiques. Au chapitre 7, nous présentons les implémentations de *computeBCov* que nous avons réalisées et les expérimentations qualitatives et quantitatives que nous avons menées. Nous terminons par l'étude des résultats de ces expérimentations.

[7] MKBEEM est l'acronyme de Multilingual Knowledge Based European Electronic Marketplace (IST-1999-10589, 1er Fév. 2000 - 1er Déc. 2002).

Enfin, nous concluons sur un bilan du travail présenté et sur quelques perspectives de recherche.

Chapitre 1

Introduction aux logiques de description

Sommaire

Les logiques de description [64] sont un formalisme de représentation de connaissances et de raisonnement étudié depuis une vingtaine d'années. Elles constituent une famille de sous-langages de la logique du premier ordre, et ont été développées pour la modélisation de structures hiérarchiques complexes et le raisonnement sur ces structures.

A l'origine, les logiques de description sont issues de la volonté de doter des formalismes graphiques de représentation de connaissances d'une sémantique formelle rigoureuse. Ceci a abouti à l'étude de fragments de la logique du premier ordre qui sont, dans leur grande majorité, décidables. De plus, l'expérience acquise en termes d'implémentation et d'optimisation assure une grande efficacité des procédures de raisonnement.

Les logiques de description permettent de décrire un domaine en deux parties : la partie intentionnelle, qualifiée de "terminologique", qui décrit les notions appelées "concepts" constituant le domaine, et la partie extensionelle qui décrit les individus qui peuplent ce domaine. Il est alors possible de raisonner sur ces descriptions du domaine. Par exemple, on peut tester la satisfiabilité d'un concept (i.e. vérifier qu'il est cohérent, non-contradictoire avec les autres concepts) et classer les concepts les uns par rapport aux autres (en utilisant le raisonnement de

subsomption). A partir de ces raisonnements de base, dits standard, de nombreux autres ont été étudiés, dits non standard, aussi bien au niveau terminologique que des individus.

Ce chapitre est divisé en trois sections. La section 1.1 présente les notions de base des logiques de description : syntaxe, sémantique et raisonnements standard de subsomption et de satisfiabilité qui en découlent. Les sections 1.2 et 1.3 présentent plus en détails trois raisonnements non standard, le plus petit subsumant commun, l'approximation et la différence, qui seront utilisés tout au long de cette thèse.

1.1 Notions de base des logiques de description

Nous présentons ici les notions de base des logiques de description utilisées par la suite dans cette thèse. Cette présentation est basée sur le Description Logic Handbook [64] ainsi que sur la thèse de Ralf Küsters [50].

1.1.1 Langage et syntaxe

Une logique de description permet de décrire l'aspect terminologique (intentionel) d'un domaine en termes de concepts, qui sont des classes d'individus, et en termes de rôles, qui sont des relations binaires entre individus. Chaque concept est décrit par une description de concept, elle-même construite par l'intermédiaire de constructeurs de concepts.

Soit N_C l'ensemble des noms de concepts et N_R l'ensemble des noms de rôles. Les descriptions de concepts élémentaires sont appelées concepts atomiques : ces concepts ne sont définis que par un nom de concept. On nomme N_A l'ensemble des concepts atomiques (on a donc $N_A \subseteq N_C$). Les descriptions complexes sont construites récursivement à partir des concepts atomiques, des rôles et des constructeurs de concepts.

Voici quelques exemples de constructeurs parmi les plus courants :

- Le concept universel $\top$ désigne l'ensemble de tous les individus du domaine.
- Le concept inconsistant $\bot$ dénote le concept vide correspondant à l'inconsistance logique.
- La négation atomique $\neg A$ d'un concept atomique A définit un concept comme le complémentaire dans $\top$ de A. Par exemple, dans le domaine de la famille, si $Feminin$ est le concept atomique qui décrit tous les individus de sexe féminin, alors $\neg Feminin$ est le concept qui décrit tous les individus qui ne sont pas de sexe féminin.
- La conjonction $\sqcap$ exprime le concept issu de l'intersection de deux concepts. Par exemple, si $Personne$ est le concept atomique décrivant tous les individus qui sont des personnes, alors $Personne \sqcap Feminin$ décrit toutes les personnes de sexe féminin, c'est-à-dire toutes les femmes.
- Les restrictions de valeur et numériques "au moins" et "au plus", notées respectivement $\forall R.C$, $\geq nR$ et $\leq nR$, expriment des restrictions sur le nombre de valeurs autorisées pour un rôle. Par exemple, $\forall aEnfant.Feminin$ décrit les individus n'ayant comme enfant que des individus de sexe féminin. La description $\geq 3aEnfant$ décrit les individus ayant au moins trois enfants.
- Le quantificateur existentiel qualifié $\exists R.C$ assure que chaque individu du concept construit est relié à un individu de C par le rôle R. Par exemple $\exists aEnfant.Feminin$ décrit les individus ayant au moins un enfant de sexe féminin.
- La disjonction $\sqcup$ décrit l'union de deux concepts. Par exemple, $Feminin \sqcup \neg Feminin$ décrit les individus qui sont de sexe féminin ou qui ne sont pas de sexe féminin, c'est-à-dire tous les individus. Cette description de concept est équivalente à $\top$.

$C, D \longrightarrow$	$\top \mid$	Concept universel
	$\bot \mid$	Concept inconsistant
	$A \mid$	Concept atomique
	$\neg A \mid$	Négation atomique
	$C \sqcap D \mid$	Intersection (ou conjonction)
	$\forall R.C \mid$	Restriction de valeur (ou quant. universel)
	$\exists R \mid$	Quantificateur existentiel
	$\geq nR \mid$	Restriction numérique "au moins"
	$\leq nR \mid$	Restriction numérique "au plus"
	$= nR \mid$	Restriction numérique exacte
	$\neg C \mid$	Négation complète
	$\exists R.C \mid$	Quantificateur existentiel qualifié
	$C \sqcup D$	Disjonction

avec A un concept atomique, C et D des descriptions de concept, R un rôle et n un entier positif ou nul.

TAB. 1.1 – Syntaxe de quelques constructeurs de concept.

– La négation complète $\neg C$ d'une description de concept C définit un concept comme le complémentaire dans $\top$ de C. Par exemple, $\neg(Personne \sqcap Feminin)$ décrit les individus qui ne sont pas des personnes de sexe féminin.

Le tableau 1.1 résume les constructeurs précédents (le constructeur $\exists R$ est équivalent à $\geq 1\ R$ et $= n\ R$ est équivalent à $(\geq n\ R) \sqcap (\leq n\ R)$).

Chaque logique de description, appelée aussi langage, est caractérisée par un et un seul ensemble de constructeurs. Le tableau 1.2 donne des exemples de langages caractérisés par les constructeurs précédents. Ce sont les langages $\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$, $\mathcal{ALEN}$ et $\mathcal{ALCN}$. Une description de concept exprimée dans la logique de description $\mathcal{L}$, c'est-à-dire en utilisant les constructeurs définissant $\mathcal{L}$, est une $\mathcal{L}$-description de concept. Par exemples, on peut construire l'$\mathcal{ALN}$-description de concept décrivant les femmes n'ayant que des filles elles-mêmes mamans d'au moins trois enfants par :

$$Personne \sqcap Feminin \sqcap \forall aEnfant.(Personne \sqcap Feminin \sqcap \geq 3\ aEnfant)$$

1.1.2 Sémantique

La volonté de développer des procédures de raisonnement rigoureuses et sans ambiguïté dans les systèmes à base de logiques de description exige l'assignation d'une sémantique formelle aux descriptions de concepts. La sémantique dénotationnelle (ou descriptive) est la plus utilisée dans ce domaine.

Informellement, étant donné un ensemble d'individus dit "univers du discours" ou "domaine d'interprétation", les concepts sont interprétés comme étant des sous-ensembles d'individus du domaine d'interprétation, et les rôles comme étant des relations binaires entre ces sous-ensembles. Ces sous-ensembles d'individus et ensembles de couples d'individus constituent les extensions des concepts et des rôles. Une fonction d'interprétation fait correspondre à chaque concept et à chaque rôle son extension.

Définition 1.1.1 (Interprétation)
Une interprétation $\mathcal{I}$ est un couple $(\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ qui est constitué d'un domaine d'interprétation non vide $\Delta^{\mathcal{I}}$ et d'une fonction d'interprétation $.^{\mathcal{I}}$ qui fait correspondre à chaque nom de concept A un ensemble $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, et à chaque rôle R une relation binaire $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

Nom et Syntaxe	Sémantique	$\mathcal{FL}_0$	$\mathcal{ALN}$	$\mathcal{ALE}$	$\mathcal{ALEN}$	$\mathcal{ALCN}$
Concept universel $\top$	$\Delta^{\mathcal{I}}$	X	X	X	X	X
Concept vide (inconsistant) $\bot$	$\emptyset$		X	X	X	X
Concept atomique A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	X	X	X	X	X
Négation atomique $\neg A$	$(\neg A)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$		X	X	X	X
Intersection (ou conjonction) $C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$	X	X	X	X	X
Restriction de valeur (ou quantificateur universel) $\forall R.C$	$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y, (x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$	X	X	X	X	X
Quantificateur existentiel $\exists R$	$(\exists R)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in R^{\mathcal{I}}\}$		X	X	X	X
Restriction numérique "au moins" $\geq nR$	$(\geq nR)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid Card(\{y, (x,y) \in R^{\mathcal{I}}\}) \geq n\}$		X		X	X
Restriction numérique "au plus" $\leq nR$	$(\leq nR)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid Card(\{y, (x,y) \in R^{\mathcal{I}}\}) \leq n\}$		X		X	X
Restriction numérique exacte $= nR$	$(= nR)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid Card(\{y, (x,y) \in R^{\mathcal{I}}\}) = n\}$		X		X	X
Quantificateur existentiel qualifié $\exists R.C$	$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in R^{\mathcal{I}} \ \wedge \ y \in C^{\mathcal{I}}\}$			X	X	X
Union (ou disjonction) $C \sqcup D$	$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$					X
Négation complète $\neg C$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$					X

avec A un concept atomique, C et D des descriptions de concept, R un rôle et n un entier positif ou nul.

TAB. 1.2 – Liste des constructeurs de concepts avec leurs nom, syntaxe et sémantique pour les langages $\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$, $\mathcal{ALEN}$ et $\mathcal{ALCN}$.

Le tableau 1.2 donne la sémantique des constructeurs vus précédemment et montre ainsi comment la fonction d'interprétation est étendue aux descriptions de concepts.

Nous donnons maintenant les définitions des notions qui découlent de la sémantique précédente, à savoir celles d'inconsistante, de satisfiabilité, de modèle, de subsomption et d'équivalence.

Définition 1.1.2 (Satisfiabilité, subsomption et équivalence)

Satisfiabilité : Un concept C dont l'interprétation $C^{\mathcal{I}}$ n'est pas vide pour au moins une interprétation $\mathcal{I}$ est dit satisfiable pour cette interprétation. On dit alors que $\mathcal{I}$ est un modèle de C. Si C n'admet pas de modèle, alors on dit que C est insatisfiable, ou inconsistant.

Subsomption : On dit que le concept D subsume le concept C, ou que C est subsumé par D, noté $C \sqsubseteq D$, si et seulement si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ pour toute interprétation $\mathcal{I}$.

Equivalence : On dit que le concept D et le concept C sont équivalents, noté $C \equiv D$, si et seulement si $C^{\mathcal{I}} = D^{\mathcal{I}}$ pour toute interprétation $\mathcal{I}$.

1.1.3 Terminologies ou TBoxes

A l'image d'un schéma relationnel qui décrit une base de données au niveau intentionnel, l'ensemble des connaissances intentionnelles d'un domaine est décrit par une terminologie, sous la forme d'axiomes terminologiques. Un axiome terminologique a deux formes possibles : $C \sqsubseteq D$ (axiome d'inclusion de concepts) et $C \equiv D$ (axiome d'égalité de concepts). Ainsi, les axiomes d'inclusion et d'égalité de concepts correspondent respectivement à des relations de subsomption et d'équivalence arbitrairement définies.

Une égalité de concepts dont le membre de gauche est un nom de concept est appelée une

Terminologie $\mathcal{T}$		
$Femme$	$\equiv$	$Personne \sqcap Feminin$
$Homme$	$\equiv$	$Personne \sqcap \neg Feminin$
$Mere$	$\equiv$	$Femme \sqcap \exists aEnfant.Personne$
$Pere$	$\equiv$	$Homme \sqcap \exists aEnfant.Personne$
$Parent$	$\equiv$	$Mere \sqcup Pere$
$GrandMere$	$\equiv$	$Femme \sqcap \exists aEnfant.Parent$
$MereGdeFamille$	$\equiv$	$Mere \sqcap \geq 3\ aEnfant$
$MereSansFille$	$\equiv$	$Mere \sqcap \forall aEnfant.\neg Femme$
$Epouse$	$\equiv$	$Femme \sqcap \exists aEpoux.Homme$

Terminologie $\mathcal{T}$ dépliée		
$Femme$	$\equiv$	$Personne \sqcap Feminin$
$Homme$	$\equiv$	$Personne \sqcap \neg Feminin$
$Mere$	$\equiv$	$Personne \sqcap Feminin \sqcap \exists aEnfant.Personne$
$Pere$	$\equiv$	$Personne \sqcap \neg Feminin \sqcap \exists aEnfant.Personne$
$Parent$	$\equiv$	$(Personne \sqcap Feminin \sqcap \exists aEnfant.Personne) \sqcup$ $(Personne \sqcap \neg Feminin \sqcap \exists aEnfant.Personne)$
$GrandMere$	$\equiv$	$Personne \sqcap Feminin \sqcap$ $\exists aEnfant.((Personne \sqcap Feminin \sqcap \exists aEnfant.Personne) \sqcup$ $(Personne \sqcap \neg Feminin \sqcap \exists aEnfant.Personne))$
$MereGdeFamille$	$\equiv$	$Personne \sqcap Feminin \sqcap \exists aEnfant.Personne \sqcap \geq 3\ aEnfant$
$MereSansFille$	$\equiv$	$Personne \sqcap Feminin \sqcap \exists aEnfant.Personne \sqcap$ $\forall aEnfant.\neg(Personne \sqcap Feminin)$
$Epouse$	$\equiv$	$Personne \sqcap Feminin \sqcap \exists aEpoux.(Personne \sqcap \neg Feminin)$

TAB. 1.3 – Un exemple de terminologie $\mathcal{T}$ constituée de 9 définitions de concepts construites à partir des concepts atomiques $Personne$ et $Feminin$, et des rôles atomiques $aEnfant$ et $aEpoux$.

définition de concept. Les définitions de concepts sont utilisées pour nommer des descriptions (complexes) de concepts. Les noms des descriptions de concepts obtenus sont appelés concepts définis et sont regroupés dans l'ensemble N_D (donc $N_C = N_A \cup N_D$, avec $N_A \cap N_D = \emptyset$).

Une terminologie $\mathcal{T}$, ou TBox (pour "terminological box"), est un ensemble d'axiomes terminologiques. Une terminologie où les descriptions manipulées sont des $\mathcal{L}$-descriptions est une $\mathcal{L}$-terminologie.

La sémantique des axiomes terminologiques et des terminologies est la suivante. Une interprétation $\mathcal{I}$ satisfait $C \sqsubseteq D$ si et seulement si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. $\mathcal{I}$ est alors un modèle de cet axiome. Une interprétation $\mathcal{I}$ satisfait $C \equiv D$ si et seulement si $C^{\mathcal{I}} = D^{\mathcal{I}}$. $\mathcal{I}$ est alors un modèle de cet axiome. Une interprétation $\mathcal{I}$ satisfait une terminologie $\mathcal{T}$ si et seulement si $\mathcal{I}$ satisfait chaque axiome de $\mathcal{T}$. $\mathcal{I}$ est alors un modèle de $\mathcal{T}$. Deux terminologies (axiomes) sont équivalentes si et seulement si elles ont les mêmes modèles.

Les terminologies que l'on utilise dans cette thèse respectent les hypothèses suivantes :

- elles ne contiennent que des définitions de concepts (pas d'axiomes d'inclusion),
- un nom de concept n'apparaît qu'une seule fois en partie gauche d'une définition,
- les terminologies sont supposées acycliques, c'est-à-dire qu'il n'existe pas de concept défini directement ou indirectement en fonction de lui-même

– et, sans perte de généralité, elles sont considérées dépliées ("unfolded" en anglais), c'est-à-dire que tous les concepts définis présents dans une description en partie droite d'une définition auront été remplacés par leur définition. Ceci implique que les parties droite des définitions ne sont exprimées qu'à l'aide des concepts atomiques et des rôles. Comme cela a été montré dans [65], le processus de dépliage peut être exponentiel dans le pire des cas (en fonction du nombre d'axiomes terminologiques et de leur taille).

Exemple 1
La terminologie $\mathcal{T}$ du tableau 1.3 décrit le domaine de la famille évoqué précédemment. On la donne d'abord dans sa forme de base, et ensuite dans sa forme dépliée. Par exemple, le concept $Mere$ est défini comme les individus qui sont des $Femme$ et qui ont au moins un enfant qui est lui-même une $Personne$. Dans sa version dépliée, ce concept s'exprime comme l'ensemble des $Personne$ qui sont des individus du sexe $Feminin$ et qui ont au moins un enfant qui est lui-même une $Personne$. ◦

Nous présentons maintenant les raisonnements terminologiques (concernant les concepts) des logiques de description. Dans la section 1.1.4, nous rappelons les définitions des raisonnements standard que sont la satisfiabilité et la subsomption. Dans la section 1.2, nous rappelons celles de deux raisonnements non standard utilisés dans cette thèse : le plus petit subsumant commun et l'approximation. Enfin, dans la section 1.3, nous rappelons et discutons de manière plus détaillée la définition et les propriétés de l'opérateur de différence sémantique de Teege [74], troisième raisonnement non standard qui joue un rôle central dans cette thèse.

1.1.4 Raisonnements standard

Il existe deux raisonnements terminologiques standard dans les logiques de description. Le premier est le test de satisfiabilité qui consiste à vérifier la cohérence logique d'un concept par rapport aux autres concepts de la terminologie considérée. Le second est le test de subsomption, entre concepts de la terminologie considérée, qui consiste à vérifier si un concept est plus spécifique qu'un autre.L'ensemble de toutes les relations de subsomption entre couples de concepts d'une terminologie constitue la hiérarchie de concepts de la terminologie. Le test de subsomption est à la base du processus de classification, appelé aussi raisonnement taxinomique, qui consiste à trouver automatiquement la position d'un concept dans une hiérarchie.

Voici la définition précise des raisonnements de satisfiabilité et de subomption [30, 64] :

– La satisfiabilité : une description de concept C est satisfiable par rapport à une terminologie $\mathcal{T}$ si et seulement si il existe un modèle $\mathcal{I}$ de $\mathcal{T}$ tel que $C^{\mathcal{I}} \neq \emptyset$. $\mathcal{I}$ est alors un modèle de C.
– La subsomption : le concept C est subsumé par le concept D selon la terminologie $\mathcal{T}$ (on note alors $C \sqsubseteq_{\mathcal{T}} D$, ou bien $\mathcal{T} \models C \sqsubseteq D$) si et seulement si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, pour tout modèle $\mathcal{I}$ de $\mathcal{T}$.

Notons que, dans la suite, nous omettrons de mentionner par rapport à quelle terminologie les relations de subsomption sont considérées parce que nous avons supposé que les descriptions étaient toujours dépliées. La subsomption se fait alors par rapport à une terminologie vide qu'il n'est pas nécessaire de mentionner.

Ces raisonnements standard des logiques de description sont bien adaptés à de nombreux problèmes, comme par exemple le traitement du langage naturel, ou les bases de données et les langages de modélisation associés. Dans le traitement du langage naturel, la sémantique des connaissances manipulées dans les logiques de description peut être utilisée dans le processus d'interprétation sémantique qui consiste à désambiguïser certaines phrases syntaxiquement am-

biguës. Dans les bases de données, on peut utiliser les raisonnements pour vérifier si certaines parties d'un schéma conceptuel sont correctes. Plus généralement, l'interaction est très profitable entre un système de gestion de bases de données, adapté au stockage et à la gestion d'un grand volume de données persistentes, et un système de représentation de connaissances et de raisonnement basé sur les logiques de description bien adapté à la gestion de la connaissance intentionnelle (le schéma de la base, les contraintes). Pour plus de détails sur ces applications et pour un panorama des autres applications des logiques de description, on se réfèrera à la partie 3 de [64].

Les algorithmes de raisonnement ont été étudiés selon deux approches : les algorithmes de subsomption structurelle et les méthodes par tableaux [30, 64].

Les algorithmes de subsomption structurelle permettent d'effectuer le test de subsomption. Ils consistent à normaliser les deux descriptions à comparer pour expliciter toutes les connaissances implicites, puis à comparer syntaxiquement les formes normales obtenues pour déduire une éventuelle relation de subsomption. Les comparaisons étant syntaxiques, ces algorithmes sont très performants en pratique. Leur limite vient de la difficulté à garantir la complétude de ce type d'algorithme en présence de constructeurs tels que la négation complète $\neg C$ ou la disjonction $\sqcup$.

Les méthodes par tableaux permettent de résoudre le test de satisfiabilité. Pour les langages fermés par rapport à la négation, elles permettent aussi de résoudre la subsomption puisqu'on a alors $(C \sqsubseteq D) \Leftrightarrow (C \sqcap \neg D$ n'est pas satisfiable). Les méthodes par tableaux consistent en la tentative de construction d'un modèle de la description dont on veut vérifier la satisfiabilité. Chaque constructeur du langage présent dans la description implique une contrainte particulière pendant la construction du modèle. Si toutes les possibilités de construction du modèle implique une contradiction entre plusieurs contraintes, alors on déduit que la description n'est pas satisfiable, sinon elle est satisfiable puisqu'alors on réussit à construire un modèle. Originellement assez peu performantes en pratique, les méthodes par tableaux sont maintenant celles qui sont utilisées dans les systèmes actuels grâce à l'utilisation conjointe de techniques d'optimisation très efficaces.

On dispose aujourd'hui d'une vision claire sur l'influence des constructeurs sur la complexité des raisonnements (satisfiabilité et subsomption) : il existe une grande variété de langages, des plus simples, ceux pour lesquels la subsomption est polynomiale en temps, aux plus complexes, ceux pour lesquels la satisfiabilité est exponentielle en temps, certains étant même indécidables (voir [64]).

Dans [20], il est montré que plus l'expressivité du langage est grande (i.e. plus on peut exprimer finement un domaine), plus la complexité des raisonnements est grande. En termes applicatifs, il faut donc chercher le meilleur compromis entre besoins en expressivité et impératifs de performances. Notons que l'expressivité d'un langage ne dépend pas uniquement du nombre de constructeurs, mais aussi de leurs interactions réciproques.

1.2 Raisonnements non standard

Les raisonnements non standard sont des raisonnements récents, proposés dans le cadre des logiques de description pour répondre à de nouveaux besoins et qui s'appuient sur les raisonnements standard des langages concernés.

Dans les sections suivantes, nous présentons les raisonnements non standard que nous utilisons dans cette thèse, à savoir :

- Le plus petit subsumant commun d'un ensemble de descriptions de concepts (ou lcs pour "least common subsumer" en anglais) : ce raisonnement consiste à trouver la plus petite description de concept (par rapport à la subsomption) qui subsume toutes celles des descriptions données en entrée (voir [6, 50, 52, 2]).
- L'approximation : ce raisonnement consiste à trouver une description la plus proche possible subsumant ou subsumée par une description donnée en entrée, exprimée en général dans un langage différent (voir [21, 22, 23, 39, 41]) ;
- la différence entre deux concepts :
  - la différence sémantique consiste à chercher les plus grandes descriptions (par rapport à la subsomption) obtenues en enlevant de la première description donnée en entrée le plus d'informations possible contenues dans la seconde (voir [74]) ;
  - la différence syntaxique consiste à chercher la plus petite description (par rapport à la taille) qui contient les parties de descriptions que la première description donnée en entrée a en plus par rapport à la seconde (voir [8, 50, 21, 23]) ;

La réécriture de concept en utilisant une terminologie [7, 8, 12, 72, 39, 42] est un autre type de raisonnement non standard. Elle consiste à reformuler un concept donné avec ceux d'une terminologie. Nous présentons son cadre général dans la section 2.1 et nous montrons aux chapitres 3 et 4 que le raisonnement auquel on s'intéresse dans cette thèse en constitue une nouvelle instance, nommée la découverte des meilleures couvertures d'un concept en utilisant une terminologie.

1.2.1 Plus petit subsumant commun (lcs)

Rechercher le plus petit subsumant commun d'un ensemble de descriptions de concepts consiste à chercher la description la plus spécifique (i.e. la plus petite par rapport à la subsomption) qui subsume toutes les descriptions données. Cette notion a été initialement définie dans [27]. Dans l'exemple 1, on a $lcs(MereSansFille, GrandMere) \equiv Mere$ ou encore $lcs(Pere, Epouse) \equiv Personne$. Intuitivement, ici, la description $Personne$ décrit les propriétés communes aux instances de $Pere$ et $Epouse$.

Définition 1.2.1 (Plus petit subsumant commun (lcs))
Soient A et B deux $\mathcal{L}$-descriptions de concepts. La $\mathcal{L}$-description de concept C est un plus petit subsumant commun de A et B si et seulement si :

- $A \sqsubseteq C$ *et* $B \sqsubseteq C$
- *C est le concept le plus spécifique (i.e. le plus petit par rapport à la subsomption) ayant cette propriété, i.e. quelle que soit la description de concept E de $\mathcal{L}$, si $A \sqsubseteq E$ et $B \sqsubseteq E$, alors $C \sqsubseteq E$.*

La définition précédente définit le lcs de deux descriptions A et B. Elle est étendue d'une manière similaire à n descriptions. Il est montré, par exemple dans [50], que le lcs n'existe pas toujours. Cependant, quand il existe et quand le constructeur $\sqcap$ est présent, il est facile de montrer qu'il est unique. C'est pourquoi on parle du (et non d'un) lcs.

Le lcs est utilisé dans des domaines comme l'apprentissage à partir d'exemples, la recherche d'informations, mais surtout pour aider à la construction et à l'utilisation de grandes bases de connaissances en logiques de description. Nous renvoyons à [50] pour un aperçu plus exhaustif des applications du lcs.

1.2.2 Approximation

L'approximation d'une description de concept C est une autre description C' qui est la plus proche possible de C mais exprimée dans un autre langage. Deux types d'approximation ont été étudiés : l'approximation faible d'un concept C consiste à trouver la description C' maximale par rapport à la subsomption et subsumée par C, et l'approximation forte d'un concept C consiste à trouver la description C' minimale par rapport à la subsomption et subsumant C.

Par exemple, considérons l'$\mathcal{ALEN}$-description suivante

$$Pere \equiv Personne \sqcap \neg Feminin \sqcap \exists aEnfant.Personne$$

L'approximation faible de ce concept dans le langage $\mathcal{ALN}$ est l'$\mathcal{ALN}$-description suivante

$$Approx(Pere) \equiv Personne \sqcap \neg Feminin \sqcap \forall aEnfant.Personne \sqcap \geq 1\ aEnfant$$

Une application de l'approximation est la traduction de base de connaissances d'un langage vers un autre, en vue d'une intégration ou d'une migration entre systèmes. Elle est aussi utile pour d'autres raisonnements non standard des logiques de description (comme par exemple dans le matching de concept [21, 23]).

L'approximation forte a été étudiée dans [21, 22, 23][8] pour effectuer l'approximation d'$\mathcal{ALC}$-descriptions dans $\mathcal{ALE}$ et pour celle d'$\mathcal{ALCN}$-descriptions dans $\mathcal{ALEN}$. L'approximation faible a été étudiée dans [39, 41] pour l'approximation d'$\mathcal{ALEN}$-descriptions dans $\mathcal{ALN}$. Dans cette thèse, nous réutilisons les résultats de [39, 41] concernant l'approximation faible, rappelés ci-dessous.

Définition 1.2.2 (Approximation d'une $\mathcal{L}_1$-description par une $\mathcal{L}_2$-description [23])
Soient $\mathcal{L}_1$ et $\mathcal{L}_2$ deux logiques de description, C une $\mathcal{L}_1$-description de concept et D une $\mathcal{L}_2$-description de concept. D est appelée une approximation faible (respectivement forte) dans $\mathcal{L}_2$ de C, notée $D = Approx_{\uparrow}(C)$ (resp. $D = Approx_{\downarrow}(C)$), si :

- *$D \sqsubseteq C$ (resp. $C \sqsubseteq D$)*
- *D est maximale (resp. minimale) par rapport à cette propriété, i.e. pour toute description D' de $\mathcal{L}_2$, $D' \sqsubseteq C$ et $D \sqsubseteq D'$ impliquent $D \equiv D'$ (resp. $C \sqsubseteq D'$ et $D' \sqsubseteq D$ impliquent $D \equiv D'$).*

Dans la suite, nous aurons besoin de calculer l'approximation faible de la négation d'une $\mathcal{ALN}$-description C de la forme $C \equiv \forall R_1.(\forall R_2.(...(\forall R_n.P)...))$ (qui est aussi une $\mathcal{ALCN}$-description) , avec P étant soit un concept atomique, soit une négation de concept atomique, soit une restriction numérique. Ainsi, $\neg C \equiv \exists R_1.(\exists R_2.(...(\exists R_n.\neg P)...))$. Les résultats de [39, 41] concernant l'approximation d'une $\mathcal{ALEN}$-description par une $\mathcal{ALN}$-description nous donnent un moyen de calculer l'approximation faible de $\neg C$. C'est l'objet du lemme suivant. Pour plus de simplicité, on utilisera la notation $C \equiv \forall R_1 R_2 ... R_n.P$ au lieu de $C \equiv \forall R_1.(\forall R_2.(...(\forall R_n.P)...))$.

Lemme 1.2.1 (Approximation faible de $\forall R_1 R_2 ... R_n.P$)
Soit une $\mathcal{ALN}$-description $C \equiv \forall R_1 R_2 ... R_n.P$ avec P un concept atomique, une négation de concept atomique ou une restriction numérique. On a :

$$\begin{aligned} Approx_{\uparrow}(\neg C) \quad = \quad & \forall R_1 R_2 ... R_n.(\neg P) \sqcap \\ & \forall R_1 R_2 ..... R_{n-1}.(\geq 1\ R_n) \sqcap \\ & \forall R_1 R_2 ..... R_{n-2}.(\geq 1\ R_{n-1}) \sqcap \\ & ... \sqcap \\ & \forall R_1.(\geq 1\ R_2) \sqcap \\ & (\geq 1\ R_1) \end{aligned}$$

[8] Le terme exact est approximation haute (pour "upper approximation" en anglais), mais c'est la même notion.

Pour une réutilisation ultérieure de l'approximation (au chapitre 4, section 4.4.2), on nomme en numérotant dans un ordre bien précis les termes de la conjonction $Approx_{\uparrow}(\neg C)$:

$$\begin{aligned}\{\text{termes de la conjonction } Approx_{\uparrow}(\neg C)\} \quad = \quad & \{c_{\uparrow}^{1} = \forall R_1 R_2 ... R_n.(\neg P),\\ & c_{\uparrow}^{2} = \forall R_1 R_2 ..... R_{n-1}.(\geq 1\ R_n),\\ & c_{\uparrow}^{3} = \forall R_1 R_2 ..... R_{n-2}.(\geq 1\ R_{n-1}),\\ & ...,\\ & c_{\uparrow}^{n} = \forall R_1.(\geq 1\ R_2),\\ & c_{\uparrow}^{n+1} = (\geq 1\ R_1)\}\end{aligned}$$

1.3 Différence

Dans cette section, nous présentons les deux opérateurs de différence existant à notre connaissance pour les logiques de descriptions. Le premier opérateur de différence est défini par Teege dans [74] selon un critère sémantique : on cherche les plus grandes descriptions B par rapport à la subsomption dont la conjonction avec D est équivalente à C. Le second est défini dans [50, 21, 23] selon un critère syntaxique : on cherche les plus petites descriptions par rapport à la taille dont la conjonction avec D est équivalente à la conjonction $C \sqcap D$.

L'intérêt des opérateurs de différence est double : ils permettent de supprimer les informations redondantes entre les descriptions (c'est le but des critères de maximalité par rapport à la subsomption et de minimalité par rapport à la taille), et ils permettent de mesurer l'écart entre deux descriptions. La suppression des redondances s'avère utile notamment dans la gestion de bases de connaissances terminologiques, en facilitant la modélisation ou la visualisation. La mesure de l'écart s'avère utile en particulier à la suite de raisonnements (non standard) qui produisent des descriptions à partir d'autres descriptions : l'opérateur syntaxique a ainsi été défini pour mesurer la qualité de l'approximation forte $\mathcal{ALE}$ d'une description $\mathcal{ALC}$ [21, 23]. L'utilisation que l'on fait de la différence s'inscrit dans le cadre d'une mesure d'éloignement entre une description et ses réécritures (voir le chapitre 2).

1.3.1 Différence sémantique

Dans cette section, nous rappelons les principaux résultats obtenus par Teege dans [74] concernant la différence sémantique de deux descriptions de concept. Nous détaillons certains d'entre eux car nous nous en servons par la suite.

Définition 1.3.1 (Opérateur de différence sémantique)
Soient C et D deux descriptions de concepts telles que $C \sqsubseteq D$. La différence (sémantique) entre C et D, notée $C - D$, est définie par :

$$C - D := Max_{\sqsubseteq}\{B | B \sqcap D \equiv C\}$$

Exemple 2
Dans l'exemple 1, on a : $GrandMere - Femme = \exists aEnfant.Parent$ qui exprime bien le fait que ce qui distingue une grand-mère d'une femme est le fait d'avoir au moins un enfant qui est un parent (i.e. qui a lui-même au moins un enfant). ◦

On peut remarquer qu'en présence du constructeur $\neg C$, la différence sémantique $C - D$ vaut toujours $\neg(D \sqcap \neg C)$ ce qui ne permet pas de déduire quoi que ce soit d'intéressant étant donné

que la réponse est exprimée avec les entrées. Cependant, en l'absence du constructeur $\neg C$, la différence présente un intérêt certain.

La définition 1.3.1 nécessite que le second argument d'une différence subsume le premier. Cependant la différence entre deux descriptions incomparables par rapport à la subsomption peut être obtenue par l'intermédiaire du plus petit subsumant commun de la manière suivante :

$$C - D := C - lcs(C, D)$$

C'est cette forme que nous utiliserons dans la suite.

Notons que, dans certaines logiques de description, le calcul de $C - D$ aboutit à un ensemble de descriptions non sémantiquement équivalentes, comme illustré dans l'exemple suivant.

Exemple 3
Soient les $\mathcal{ALN}$-descriptions C et D : $C \equiv \forall R.\bot$ et $D \equiv (\forall R.P') \sqcap (\forall R.(\leq 4S))$. Les deux $\mathcal{ALN}$-descriptions $(\forall R.\neg P')$ et $(\forall R.(\geq 5S))$ sont les solutions de la différence $C - D$. Ces deux descriptions ne sont pas équivalentes. ◦

Teege [74] donne une condition suffisante pour caractériser les langages dans lesquels la différence sémantique est toujours unique (i.e. toutes les descriptions résultant d'une différence sont sémantiquement équivalentes) et peut être implémentée d'une manière syntaxique simple par un opérateur de différence ensembliste entre ensemble de termes de conjonctions, appelés clauses. Avant d'énoncer précisément ces propriétés, nous donnons les définitions des notions utilisées. En particulier, la notion de "clause" définie par Teege, qui désigne habituellement une disjonction de littéraux, désigne ici un composant conjonctif élémentaire, autrement dit un terme d'une conjonction qui ne peut être lui-même décomposé en une conjonction non triviale.

Définition 1.3.2 (Clause, forme clausale et équivalence structurelle)
Soit une logique de description $\mathcal{L}$.

- *Une clause dans $\mathcal{L}$ est une description A qui respecte la propriété suivante :*

  $$(A \equiv B \sqcap A') \Rightarrow (B \equiv \top) \vee (B \equiv A) \vee (A' \equiv \top) \vee (A' \equiv A)$$

  Chaque conjonction $A_1 \sqcap \ldots \sqcap A_n$ de clauses peut être représentée par l'ensemble de clauses $\{A_1, \ldots, A_n\}$.
- *Un ensemble de clauses $A = \{A_1, \ldots, A_n\}$ est dit réduit si :*
  - *soit $n = 1$,*
  - *soit aucune clause ne subsume la conjonction des autres clauses, i.e. :*

    $$\forall i \mid i \in \{1, .., n\},\ A_i \not\sqsupseteq (A \setminus A_i)$$

    (avec $A \setminus A_i$ étant un raccourci pour dénoter la conjonction de toutes les clauses de A sauf A_i). L'ensemble A est alors une forme clausale réduite (ou RCF en anglais pour "reduced clause form") de la description $B \equiv A_1 \sqcap \ldots \sqcap A_n$.
- *Soit $A = \{A_1, \ldots, A_n\}$ et $B = \{B_1, \ldots, B_m\}$ deux ensembles de clauses réduits de $\mathcal{L}$ (i.e. deux RCF de deux $\mathcal{L}$-descriptions). A et B sont structurellement équivalents (on note $A \cong B$) si et seulement si : $n = m$ et $\forall i \in \{1, .., n\}, \exists (j, k) \in \{1, .., n\}^2 \mid A_i \equiv B_j$ et $B_i \equiv A_k$.*
- *Une logique de description $\mathcal{L}$ est dite à RCF structurellement uniques si toutes les RCF associées à une description sont structurellement équivalentes.*

Exemple 4
Soit le langage $\mathcal{FL}_0 \cup (\geq nR)$. Soit le concept A défini de la manière suivante :

$$A \equiv \forall R_1.P_1 \sqcap \forall R_2.(P_2 \sqcap \forall R_3. \geq 2R_4) \sqcap \forall R_2.\forall R_3. \geq 1R_4$$

L'ensemble des clauses de A est le suivant :

$$\{\forall R_1.P_1, \forall R_2.P_2, \forall R_2.\forall R_3. \geq 2R_4, \forall R_2.\forall R_3. \geq 1R_4\}$$

Comme on a :

$$\forall R_1.P_1 \sqcap \forall R_2.(P_2 \sqcap \forall R_3. \geq 2R_4) \sqsubseteq \forall R_2.\forall R_3. \geq 1R_4$$

alors la RCF de A est l'ensemble de clauses :

$$\{\forall R_1.P_1, \forall R_2.P_2, \forall R_2.\forall R_3. \geq 2R_4\}$$

∘

Le théorème suivant montre que, dans les logiques de description à RCF structurellement uniques, le calcul de la différence sémantique peut être réalisé par l'opérateur de différence ensembliste entre ensembles de clauses. Cet opérateur est noté $\setminus_{\equiv}$. De plus, dans ce cas, le résultat de la différence est alors unique (modulo l'équivalence).

Théorème 1.3.1 (Différence pour les langages à RCF structurellement uniques)
Soit $\mathcal{L}$ une logique de description à RCF structurellement uniques. Soient C et D des $\mathcal{L}$-descriptions données par leur RCF, avec $C \sqsubseteq D$. On a alors :

- *$C - D$ est sémantiquement unique (une seule description en résultat, modulo l'équivalence)*
- *et $C - D$ peut être calculée par l'opérateur de différence structurelle, i.e. $C - D = C \setminus_{\equiv} D$.*

Dans [74], Teege définit la propriété de subsomption structurelle, rappelée ci-dessous. Cette propriété est une condition suffisante pour qu'un langage soit à RCF structurellement uniques et donc possède une différence sémantique unique et calculable facilement par l'opérateur $\setminus_{\equiv}$. Ces résultats sont rappelés maintenant.

Définition 1.3.3 (Propriété de subsomption structurelle)
Une logique de description $\mathcal{L}$ est dite ayant la propriété de subsomption structurelle si et seulement si pour toute clause A de $\mathcal{L}$ et toute description $B = B_1 \sqcap \ldots \sqcap B_m$ de $\mathcal{L}$ donnée par une RCF, on a :

$$A \sqsupseteq B \Leftrightarrow \exists 1 \leq i \leq m \mid A \sqsupseteq B_i$$

Le théorème suivant dit que si $\mathcal{L}$ est un langage qui possède la propriété de subsomption structurelle, alors $\mathcal{L}$ est à RCF structurellement uniques.

Théorème 1.3.2 (Subsomption structurelle et RCF structurellement uniques)
Soit $\mathcal{L}$ une logique de description ayant la propriété de subsomption structurelle. Soient A et B deux RCF de deux $\mathcal{L}$-descriptions. On a :

$$A \equiv B \Rightarrow A \text{ et } B \text{ sont structurellement équivalentes}$$

D'après le théorème précédent, la différence sémantique dans un langage ayant la propriété de subsomption structurelle peut être calculée avec l'opérateur de différence $\setminus_{\equiv}$.

Existence du lcs Pour les langages à subsomption structurelle, on montre que l'existence du lcs entre deux clauses quelconques assure l'existence du lcs entre deux descriptions quelconques (voir le lemme A.1.2 page 139). Cependant, la notion de clause ne permet pas d'être plus précis quant à l'existence du lcs pour les langages ayant la propriété de subsomption structurelle.

Pour déterminer si le lcs existe pour ces langages ou seulement une sous-famille (et laquelle), on peut utiliser les travaux de [27] qui étudient les liens entre le lcs et les langages à subsomption structurelle. Il est important de préciser que la subsomption structurelle définie dans [27] n'est pas la même que la propriété de subsomption structurelle utilisée ici (voir la définition 1.3.3 ci-dessus). Cependant, cette dernière étant plus générale que celle de [27], on peut réutiliser certains des résultats de [27]. En particulier, une condition nécessaire d'existence du lcs en fonction de la nature des constructeurs considérés est donnée. Cette condition dit aussi comment calculer le lcs avec ces constructeurs.

Ainsi, on fait l'hypothèse que les langages qui ont la propriété de subsomption structurelle et qui sont utilisés dans cette thèse possèdent des constructeurs qui respectent la condition donnée dans [27], ce qui assure que le lcs existe et que l'on sait comment le calculer.

D'une manière générale, cette hypothèse n'est pas très restrictive. En effet, les cas où le lcs n'existe pas ne sont pas nombreux. Dans [50] (section 3.1.1), Küsters énumère les (rares) cas connus dans lesquels le lcs peut ne pas exister. Le premier cas est celui où il n'y a pas de subsumant commun, c'est-à-dire où $\top$ n'appartient pas au langage. Le second cas est celui où $\sqcap$ n'appartient pas au langage. Le troisième est celui où il existe une chaîne infinie de descriptions subsumées les unes par les autres telle que $C_1 \sqsupset C_2 \sqsupset C_3 \sqsupset ...$, et telle que chaque C_i subsume les descriptions dont on cherche le lcs.

1.3.2 Algorithme et propriété de subsomption structurelle

Même si elles sont très proches, les notions de "propriété" (définition 1.3.3) et d'"algorithme" (section 1.1.4) de subsomption structurelle sont distinctes. En effet, un langage qui a la propriété de subsomption structurelle possède obligatoirement un algorithme de subsomption structurelle (l'étape de normalisation est alors constituée par l'obtention d'une RCF de B, si on teste $B \sqsubseteq A$). Cependant la réciproque n'est pas vérifiée : l'existence d'un algorithme de subsomption structurelle n'implique pas que le langage concerné possède la propriété de subsomption structurelle. Par exemple, il existe des algorithmes de subsomption structurelle pour $\mathcal{ALN}$, mais ce langage ne vérifie pas la propriété de subsomption structurelle. En résumé, pour un langage donné, avoir la propriété de subsomption structurelle est une condition plus forte qu'avoir un algorithme de subsomption structurelle.

Parmi les logiques qui ont la propriété de subsomption structurelle, il y a par exemple $\mathcal{FL}_0$, $\mathcal{FL}_0 \cup (\geq n\, R)$ ou $\mathcal{L}_1$. $\mathcal{L}_1$ est définie dans [74] de la manière suivante[9] :

- $\sqcap, \sqcup, \top, \bot, (\geq n\, R), (\exists R.C)$ et $(\exists f.C)$ pour les descriptions de concepts (avec C un concept, R un rôle et f un rôle fonctionnel ou attribut),
- $\bot, \circ, |$ pour les descriptions de rôles,
- $\bot, \circ$ pour les rôles fonctionnels.

En termes de complexité, il est clair que pour les langages ayant la propriété de subsomption structurelle, la complexité du calcul de la différence en lui-même est polynomiale en le nombre de clauses des RCF des descriptions courantes. La complexité globale est donc celle de la subsomption du langage correspondant (pour les tests d'équivalence entre clauses). Par exemple, pour $\mathcal{FL}_0$, comme obtenir une RCF pour une description est polynomial en fonction de la taille

[9] Pour la sémantique des constructeurs de $\mathcal{L}_1$, se référer à l'annexe A.1.1 page 138.

de la description et aboutit à un nombre polynomial de clauses, et comme la subsomption est polynomiale, alors le calcul complet de la différence est polynomial aussi. Il n'y a pas d'autres résultats de complexité dans [74].

1.3.3 Différence syntaxique

Récemment, un deuxième opérateur de différence a été proposé : la différence syntaxique [50, 21, 23]. La différence syntaxique C - D a pour but de supprimer les parties, ou "sous-descriptions", de descriptions de C qui sont aussi présentes dans D. On dit que cet opérateur élimine les redondances syntaxiques, ce qui a pour conséquence l'obtention de descriptions minimales en taille.

Intuitivement, D est une sous-description de C, noté $D \preceq_d C$, si on peut obtenir D à partir de C en enlevant des termes des conjonctions ou disjonctions présentes dans C, en remplaçant des parties de descriptions de C par $\bot$ ou par des sous-descriptions de ces parties (qui sont elles-mêmes des descriptions). Cet ordre, basé sur la notion de sous-description et défini dans [8, 50, 21, 23], permet de définir la différence syntaxique. $\mathcal{ALE}$ et $\mathcal{ALC}$ sont les seuls langages à notre connaissance pour lesquels la différence syntaxique a été étudiée.

Définition 1.3.4 (Différence syntaxique dans $\mathcal{ALC}$ et $\mathcal{ALE}$)
Soient C une $\mathcal{ALC}$-description de concept et D une $\mathcal{ALE}$-description de concept. La différence (syntaxique) $C - D$ de C et D est définie comme une description de concept $\mathcal{ALC}$ E minimale par rapport à $\preceq_d$ telle que $E \sqcap D \equiv C \sqcap D$.

Dans l'exemple 1, on a : $Epouse - GrandMere = \exists aEpoux.Homme$. La raison est que $Epouse \sqcap GrandMere \equiv Femme \sqcap \exists aEnfant.Parent \sqcap \exists aEpoux.Homme$, et $\exists aEpoux.Homme$ est la description minimale par rapport à $\preceq_d$ telle que
$\exists aEpoux.Homme \sqcap Femme \sqcap \exists aEnfant.Parent \equiv Epouse \sqcap GrandMere$.

1.3.4 Discussion sur le choix de l'opérateur de différence

L'étude présentée dans l'annexe A.2.2 page 142 pour le langage $\mathcal{ALN}$ montre que les deux opérateurs sont très proches : si on définit la différence syntaxique pour $\mathcal{ALN}$ de la même façon qu'elle l'est pour $\mathcal{ALE}$ et $\mathcal{ALC}$, on constate que ces deux différences se distinguent seulement par le traitement de l'inconsistance, et ce uniquement dans deux cas particuliers (donnés dans l'exemple ci-dessous). Or, dans $\mathcal{ALN}$, les cas de résultats multiples sont uniquement issus des décompositions de $\bot$ (voir le calcul de la différence sémantique dans $\mathcal{ALN}$ au théorème 4.2.2 page 52). La différence syntaxique a donc l'avantage de produire un résultat unique dans $\mathcal{ALN}$.

Exemple 5
Soient les $\mathcal{ALN}$-descriptions $C \equiv \forall R.\bot$ et $D \equiv \forall R.(\neg P \sqcap P')$. On a :
Pour la différence sémantique : $C - D = \{\forall R.P, \forall R.\neg P'\}$.
Pour la différence syntaxique : $C - D = \forall R.\bot$.
Ce cas particulier a déjà été évoqué dans [50], et la conclusion est la suivante : s'il est vrai que la différence sémantique donne effectivement l'écart sémantique, c'est-à-dire ce qu'il faut ajouter par conjonction à D pour obtenir C, les résultats multiples sont moins faciles à appréhender par un utilisateur, par exemple un ingénieur des connaissances, que le résultat unique de l'opérateur de différence syntaxique. De plus, dans ce cas, on peut estimer que le résultat de l'opérateur syntaxique est plus intuitif car il ne fait intervenir aucune décomposition de $\bot$.
Soient les $\mathcal{ALN}$-descriptions $C \equiv \forall R.P$ et $D \equiv \forall R.\neg P$. On a :
Pour la différence sémantique : $C - lcs(C, D) = \{\forall R.P\}$.

Pour la différence syntaxique : $C - D = \forall R.\bot$.
Dans ce cas, le résultat de l'opérateur sémantique apparaît plus intuitif. En effet, obtenir C en résultat de $C - D$, c'est dire qu'il n'y a aucun point commun entre C et D, ce qui est vérifié dans ce cas puisque le lcs de C et D est $\top$. Par contre le résultat de l'opérateur syntaxique est plus difficile à interpréter, car il ne s'apparente ni à ce qui reste de C quand on lui a enlevé D, ni à ce qu'il faut ajouter à D pour obtenir C. ◦

Selon le contexte applicatif dans lequel est utilisée la différence, et selon la manière dont on veut exploiter les résultats obtenus, on choisira l'un ou l'autre des deux opérateurs. Aucun des deux n'est meilleur a priori. Comme on le verra au chapitre 2 de cette thèse, nous calculons la différence de plusieurs descriptions de concepts avec une même description, et selon la nature des différences obtenues, nous classons les descriptions les unes par rapport aux autres. Nous pensons que la différence sémantique est plus adaptée dans ce contexte en raison de son traitement des inconsistances qui, d'après les exemples précédents, est plus cohérent et plus général. En effet, la différence sémantique donne plus d'informations que la différence syntaxique puisque les résultats de l'opérateur sémantique subsument ceux de l'opérateur syntaxique. Ceci implique qu'on calcule de manière plus complète tous les aspects de la différence entre deux descriptions. On peut donc utiliser les résultats de la différence dans des cas plus nombreux. La contrepartie est que la différence sémantique peut générer des résultats plus difficiles à manipuler (notamment par l'utilisateur car ils plus complexes). Cependant, on rappelle qu'en dehors de ces cas d'inconsistance, les deux opérateurs donnent les mêmes résultats.

Chapitre 2

Une nouvelle réécriture de concept en utilisant une terminologie : les meilleures couvertures

Sommaire

Nous présentons maintenant l'objet de cette thèse qui est l'étude d'une nouvelle instance de la réécriture de concept en utilisant une terminologie. Nous l'avons nommée *la découverte des meilleures couvertures d'un concept en utilisant une terminologie*. Comme nous l'avons évoqué précédemment, la réécriture de concept en utilisant une terminologie est une famille de raisonnements non standard pour les logiques de description. Son cadre est défini dans [7, 8].

Nous rappelons que l'étude de cette nouvelle réécriture est motivée par le problème de la découverte dynamique de services web sémantiques. Nous montrons que les besoins induits en termes de flexibilité ne correspondent pas aux réécritures existantes basées sur l'équivalence ou la subsomption. Nous justifions ainsi l'étude d'une réécriture définie par un critère plus souple consistant en la maximisation des informations communes entre le concept à récrire et les réécritures par la minimisation de leurs différences.

Dans la section 2.1, nous rappelons le cadre formel de la réécriture dans les logiques de description et ses instances existantes. Puis, dans la section 2.2, nous donnons l'idée intuitive des meilleures couvertures. Enfin, dans la section 2.3, nous précisons pour quels langages nous étudions les meilleures couvertures.

Pour la présentation et la motivation des meilleures couvertures dans ce chapitre, nous sommes amenés à évoquer les réécritures existantes. Cependant, c'est l'objet du chapitre 5 que de détailler l'état de l'art des réécritures.

2.1 Cadre général

Intuitivement, le problème de la réécriture de concept en utilisant une terminologie est le suivant : étant données une terminologie $\mathcal{T}$ et une description de concept Q, peut-on récrire Q

en une description E *reliée au mieux* à Q mais exprimée avec des concepts définis de $\mathcal{T}$? D'après le cadre général de la réécriture défini dans [7, 8] et rappelé ci-dessous, il s'agit de construire E à partir de Q et d'une terminologie $\mathcal{T}$, en sachant que Q, E et $\mathcal{T}$ peuvent être exprimées dans des langages différents, qu'il doit y avoir une relation précise entre Q et E, qui identifie les réécritures valides, et que E doit minimiser un ordre, qui identifie les meilleures réécritures.

Définition 2.1.1 (Cadre général de la réécriture)
Soient trois langages $\mathcal{L}_d$ (d pour destination), $\mathcal{L}_s$ (s pour source) et $\mathcal{L}_t$ (t pour terminologie). Un problème de réécriture est donné par :

- *une $\mathcal{L}_t$ -terminologie $\mathcal{T}$, dont l'ensemble des concepts définis est noté $\mathcal{S}_\mathcal{T}$,*
- *une $\mathcal{L}_s$-description de concept Q et*
- *une relation binaire $\rho \subseteq \mathcal{L}_s \times \mathcal{L}_d$ entre des $\mathcal{L}_s$- et $\mathcal{L}_d$-descriptions.*

Une réécriture (valide) de Q dans $\mathcal{L}_d$ en utilisant $\mathcal{T}$ est une $\mathcal{L}_d$-description de concept E construite en utilisant des concepts définis de $\mathcal{T}$ (i.e. appartenant à $\mathcal{S}_\mathcal{T}$) telle que $Q \rho E$ est vérifié.

Etant donné un ordre $\preceq$ sur les $\mathcal{L}_d$-descriptions de concepts, une réécriture E est dite minimale par rapport à $\preceq$ s'il n'existe pas de réécriture E' telle que $E' \preceq E$ et $E' \not\succeq E$. Cet ordre définit le critère d'optimalité qui distingue les meilleures réécritures.

Il existe deux instances de ce cadre de la réécriture dans les logiques de description. Elles se distinguent par leur domaine d'application, les langages concernés et surtout par la relation entre le concept à récrire et les réécritures, qui est soit l'équivalence, soit la subsomption. La première instance est la *réécriture équivalente de taille minimale*, ou *réécriture minimalement équivalente* [9] : son but est de récrire un concept en un autre équivalent (la relation ρ est l'équivalence) mais syntaxiquement plus petit (l'ordre $\preceq$ est l'ordre sur les tailles des réécritures). Cette réécriture est utile par exemple dans le domaine de la gestion et de l'intégration de bases de connaissances où un ingénieur des connaissances peut s'en servir pour rendre plus lisibles les connaissances qu'il a à gérer (la description d'un concept pouvant parfois atteindre plusieurs pages de texte). La seconde instance est la *réécriture maximalement contenue* [12, 72, 39, 42] : son but est de récrire un concept en un autre, sachant que le concept à récrire doit subsumer la réécriture (ρ est la subsomption $\sqsubseteq$) et que la réécriture doit être maximale par rapport à la subsomption ($\preceq$ est la subsomption inverse $\sqsupseteq$). Cette réécriture a été utilisée dans le domaine de l'intégration des systèmes d'informations pour résoudre le problème de répondre à des requêtes en utilisant des vues [44]. Nous présentons maintenant la nouvelle instance du cadre de la réécriture qui est étudiée dans cette thèse.

2.2 Meilleures couvertures : présentation et motivations

Soient une terminologie $\mathcal{T}$ et une description Q. On cherche les réécritures E de Q qui se rapprochent le plus de Q. Plus précisément, le concept à récrire et les réécritures doivent partager au moins une information, et leur proximité sémantique est définie (i) en maximisant la quantité de leurs informations communes et (ii) en minimisant la quantité des informations spécifiques aux réécritures. Pour cela, on cherche les réécritures E qui (i) minimisent la différence $Q - E$, et (ii) minimisent la différence $E - Q$. $Q - E$ est nommée le *rest*, et $E - Q$ est nommée le *miss*. Le rest est ce qui reste de Q une fois ses points communs avec les réécritures E enlevés. Le miss est ce qui manque à Q par rapport à ses réécritures E, c'est-à-dire les informations spécifiques contenues dans E, une fois les points communs avec Q otés. La figure 2.1 illustre intuitivement cette approche.

Cette réécriture constitue une nouvelle instance du cadre formel précédent :

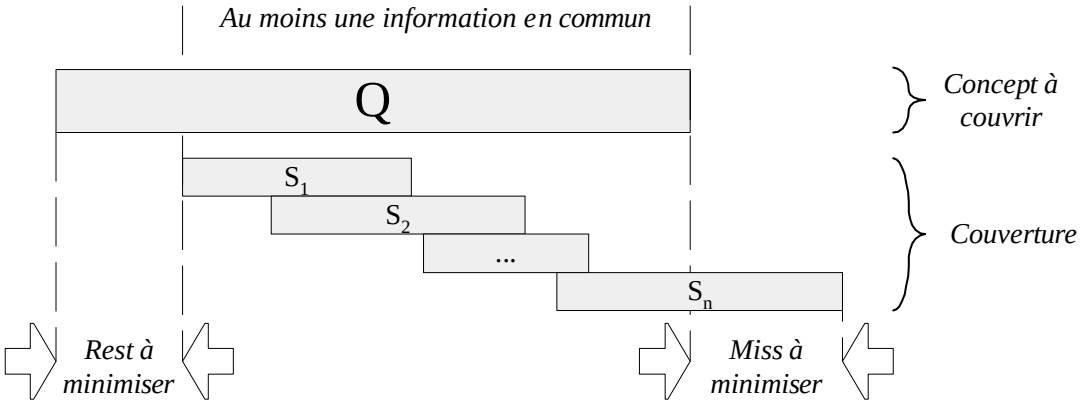


FIG. 2.1 – Représentation intuitive du principe des meilleures couvertures.

$Trajet \equiv$	$Service \sqcap \forall typeService.VoyageAerien \sqcap \forall vol\_Aller.Lieu \sqcap$ $\forall vol\_Retour.Lieu \sqcap \forall compagnieAerienne.Compagnie\_Italienne$
$Hotel \equiv$	$Service \sqcap \forall typeService.Hebergement \sqcap \forall typeHeberg.Hotellerie$
$Hotel\_Luxe \equiv$	$Service \sqcap \forall typeService.Hebergement \sqcap \forall typeHeberg.Hotellerie \sqcap$ $\forall equipement.Piscine$
$Voiture \equiv$	$Service \sqcap \forall typeService.LocationVehicule \sqcap \forall typeVehicule.Berline \sqcap$ $\forall constructeur.CompagnieAllemande \sqcap \forall equipement.BoiteAuto$
$Pack\_Voyage \equiv$	$Service \sqcap \forall vol\_Aller.Lieu \sqcap$ $\forall compagnieAerienne.Compagnie\_Francaise \sqcap$ $\forall typeHeberg.Hotellerie \sqcap \forall typeVehicule.Berline \sqcap \forall equipement.4\times4$

TAB. 2.1 – Une terminologie $\mathcal{T}$ et un concept $Q = Pack\_Voyage$ à couvrir, exprimés dans la logique de description $\mathcal{FL}_0$.

- La relation ρ, définissant une réécriture valide, est donnée par le fait que Q et E doivent partager au moins une information non triviale. Cela se traduit par un rest différent de Q, soit $Q - E \neq Q$.
- L'ordre $\preceq$, définissant les meilleures réécritures, est donné par le fait de maximiser la proximité sémantique entre Q et E. La maximisation des informations communes entre Q et E se traduit par la minimisation du rest (pour qu'il contienne le moins d'informations possible). La minimisation des informations spécifiques aux réécritures se traduit par la minimisation du miss.

La minimisation des rest et miss signifie qu'on cherche des rest et miss contenant le moins possible d'informations. Nous étudions par la suite deux critères marquant cette quantité minimal d'informations. Le premier est la minimalité par rapport à la taille des rest et miss. Le second est la maximalité par rapport à la subsomption. Le choix d'un critère ou de l'autre sera justifié au cas par cas dans les chapitres 3 et 4.

Exemple 6

Soient la terminologie $\mathcal{T}$ et la requête $Q = Pack\_Voyage$ données dans le tableau 2.1 page 23. Une couverture étant une conjonction de concepts de $\mathcal{T}$, comme $\mathcal{T}$ possède 4 concepts définis il y a $2^4 = 16$ couvertures E possibles dans cet exemple. Après leur examen, on trouve seulement

deux E qui maximisent les informations communes avec Q, c'est-à-dire qui minimisent leur rest. Ce sont :

$$Couv_1 = Trajet \sqcap Hotel \sqcap Voiture \text{ et } Couv_2 = Trajet \sqcap Hotel\_Luxe \sqcap Voiture.$$

En effet, on obtient par différence sémantique :

$$\begin{array}{ll} \text{rest de } Couv_1 & = Pack\_Voyage - Couv_1 \\ & = \forall compagnieAerienne.Compagnie\_Francaise \sqcap \forall equipement.4{\times}4 \\ & \text{et} \\ \text{rest de } Couv_2 & = Pack\_Voyage - Couv_2 \\ & = \forall compagnieAerienne.Compagnie\_Francaise \sqcap \forall equipement.4{\times}4 \end{array}$$

Ceci signifie que les deux couvertures couvrent $Pack\_Voyage$ de la même façon, c'est-à-dire que les informations communes entre $Pack\_Voyage$ et $Couv_1$ d'une part, et entre $Pack\_Voyage$ et $Couv_2$ d'autre part sont les mêmes. De plus, il est clair qu'on ne peut pas couvrir $Pack\_Voyage$ plus que cela, c'est-à-dire qu'on ne peut pas trouver d'autres conjonctions de services qui aient plus d'informations en commun que $Couv_1$ ou $Couv_2$ avec $Pack\_Voyage$.

Parmi $Couv_1$ et $Couv_2$, seule $Couv_1$ minimise le miss. Ceci est prouvé par le calcul des miss suivants :

$$\begin{array}{ll} \text{miss de } Couv_1 & = Couv_1 - Pack\_Voyage \\ & = \forall typeService.VoyageAerien \sqcap \forall vol\_Retour.Lieu \sqcap \\ & \forall compagnieAerienne.Compagnie\_Italienne \sqcap \\ & \forall typeService.Hebergement \sqcap \forall typeService.LocationVehicule \sqcap \\ & \forall constructeur.CompagnieAllemande \sqcap \forall equipement.BoiteAuto \\ \text{miss de } Couv_2 & = Couv_2 - Pack\_Voyage \\ & = \forall typeService.VoyageAerien \ \sqcap \ \forall vol\_Retour.Lieu \sqcap \\ & \forall compagnieAerienne.Compagnie\_Italienne \sqcap \\ & \forall typeService.Hebergement \sqcap \forall equipement.Piscine \ \sqcap \\ & \forall typeService.LocationVehicule \ \sqcap \\ & \forall constructeur.CompagnieAllemande \sqcap \forall equipement.BoiteAuto \end{array}$$

On déduit que la taille du miss de $Couv_1$ est plus petite que la taille du miss de $Couv_2$ (car $Hebergement\_Luxe$ ajoute à $Couv_2$ l'information sur la présence d'une piscine).
En résumé, $Couv_1$ et $Couv_2$ sont les couvertures de $Pack\_Voyage$ en utilisant $\mathcal{T}$, et $Couv_1$ est la meilleure couverture de $Pack\_Voyage$ en utilisant $\mathcal{T}$. ∘

La relation ρ des meilleures couvertures définit une réécriture plus générale que celles basées sur l'équivalence ou la subsomption. En effet, en cas d'existence de réécritures équivalentes ou subsumées, ces dernières sont trouvées préférentiellement puisqu'elles maximisent par nature les informations communes avec Q. Par contre, en cas d'absence de telles réécritures, les meilleures couvertures sont recherchées et peuvent aboutir à des solutions là où les réécritures basées sur l'équivalence et la subsomption ne renvoient aucun résultat. Ainsi seuls les concepts n'ayant strictement aucun point commun avec la terminologie utilisée ne donnent aucun résultat à la fin de la recherche des meilleures couvertures.

Dans l'exemple précédent, on peut voir qu'aucune conjonction de concepts de la terminologie n'est équivalente avec $Pack\_Voyage$ (car, par exemple, aucun concept ne contient d'information concernant une compagnie aérienne). Il n'y a donc pas de réécriture équivalente. De même, il

n'y a pas de réécriture maximalement contenue dans $Pack\_Voyage$. Les meilleures couvertures produisent tout de même un résultat, prouvant ainsi leur plus grande flexibilité par rapport aux réécritures précédentes.

Pour caractériser la notion de plus grande proximité entre le concept à récrire Q et les couvertures E, on aurait pu penser à maximiser le lcs de Q et E pour maximiser leurs informations communes. Cependant, par rapport à la minimisation des rest et miss, cette approche a deux inconvénients :

- On ne départage pas les couvertures E sur la base des informations supplémentaires qu'elles contiennent (c'est le rôle de la minimisation du miss). Ainsi, les couvertures sont moins finement distinguées les unes par rapport aux autres.
- On ne calcule pas explicitement les différences (le rest et le miss) entre Q et E. Or, il peut être intéressant d'avoir ces descriptions de concepts. On peut en effet les réutiliser en entrées d'autres raisonnements ou les exploiter de diverses manières. Par exemple, dans le contexte de la découverte de services, on montre au chapitre 6 que le miss peut servir à établir un dialogue entre le système et l'utilisateur pour qu'il puisse préciser sa requête. De même, le rest peut être envoyé en entrée d'autres systèmes afin de trouver d'autres services complétant les services déjà découverts.

Il est important de remarquer que, dans la définition des meilleures couvertures, les rest et miss ont des rôles différents. Le rest est l'élément le plus important : il définit les couvertures valides puisqu'il caractérise les informations communes avec Q. Le miss, quant à lui, définit une fonction de coût associée aux couvertures. Il sert à sélectionner, parmi les couvertures maximisant les informations communes, celles de meilleur coût. Il y a deux raisons au choix d'un miss contenant les informations présentes dans E et absentes de Q. La première est, comme on l'a dit précédemment, que minimiser ce miss revient à assurer la proximité de E et Q. La seconde provient du contexte applicatif dans lequel a débuté cette étude, c'est-à-dire le projet MKBEEM (voir la section 6.2.1 page 106). Comme dans MKBEEM, le miss sert à établir un dialogue avec l'utilisateur pour qu'il complète sa requête, sa minimisation revient à alléger ce dialogue. C'est-à-dire que plus le miss contient d'informations, plus le dialogue entre le système et l'utilisateur est long. Or on veut limiter le nombre d'interactions système - utilisateur. Le coût porté par le miss est donc dans ce cas la longueur du dialogue qu'il induit, longueur qui est proportionnelle à sa taille.

2.3 Langages étudiés

Pour étudier des solutions au problème, il faut choisir les langages source, destination et de terminologie. Comme on le suggère dans l'exemple précédent, chaque réécriture représente une combinaison particulière de certains concepts définis S_i. Cela revient à chercher des conjonctions de concepts définis S_i. Ainsi, à l'instar de certaines instances de la réécriture maximalement contenue, le langage destination est $\{\sqcap, \sqcup\}$, $\sqcap$ pour les conjonctions de S_i, et $\sqcup$ pour autoriser plusieurs conjonctions différentes.

Le choix des langages source et de terminologie est guidé par le choix de l'opérateur de différence. On a choisi l'opérateur de différence sémantique de Teege pour les raisons données dans la section 1.3.4. Celui-ci peut être implémenté de manière facile par une différence ensembliste si le langage étudié est à RCF structurellement uniques. On choisit donc de mener, dans un premier temps, l'étude des meilleures couvertures pour des langages qui ont la propriété de subsomption

structurelle (qui assure aux RCF d'être structurellement uniques). Cette étude est présentée au chapitre 3.

Dans un second temps, on mène l'étude avec le langage $\mathcal{ALN}$ pour les langages source et destination. En effet $\mathcal{ALN}$ est connu pour offrir un bon compromis entre expressivité et complexité de raisonnement. Cependant, $\mathcal{ALN}$ permet l'obtention de l'inconsistance de manière non triviale. Ceci implique que les cas d'inconsistances doivent être découverts et que la différence sémantique dans $\mathcal{ALN}$ est plus complexe à calculer. L'étude des meilleures couvertures pour le langage $\mathcal{ALN}$ est donc plus complexe que pour les langages ayant la propriété de subsomption structurelle. Elle fait l'objet du chapitre 4.

Chapitre 3

Une solution pour les langages ayant la propriété de subsomption structurelle

Sommaire

Dans ce chapitre, nous étudions le problème de la découverte des meilleures couvertures d'un concept en utilisant une terminologie, dans le cadre des logiques de description ayant la propriété de subsomption structurelle.

Dans la section 3.1, nous définissons ce problème, que l'on nomme $\mathcal{BCOV}(\mathcal{T}, Q)$. Nous montrons qu'il est équivalent à la recherche des transversaux minimaux de coût minimal dans un hypergraphe. Nous donnons ensuite l'algorithme *computeBCov* pour calculer les meilleures couvertures dans la section 3.2. Enfin dans la section 3.3, nous étudions la complexité de $\mathcal{BCOV}(\mathcal{T}, Q)$ et montrons que c'est un problème NP-Difficile.

Dans les exemples du chapitre, nous utilisons le langage $\mathcal{FL}_0$ qui a la propriété de subsomption structurelle.

3.1 Etude du problème $\mathcal{BCOV}(\mathcal{T}, Q)$

Dans cette section, nous définissons notre problème et montrons comment il peut être modélisé et résolu en termes d'hypergraphes.

3.1.1 Définition du problème $\mathcal{BCOV}(\mathcal{T}, Q)$

Introduisons quelques notations afin de pouvoir formellement définir le problème de découverte des meilleures couvertures d'un concept en utilisant une terminologie pour les langages ayant la propriété de subsomption structurelle.

Soit $\mathcal{L}$ une logique de description ayant la propriété de subsomption structurelle, $\mathcal{T}$ une $\mathcal{L}$-terminologie acyclique, et $Q \not\equiv \bot$ une $\mathcal{L}$-description de concept. L'ensemble des concepts définis de $\mathcal{T}$ est noté $\mathcal{S}_\mathcal{T} = \{S_i, i \in \{1, .., n\}\}$ avec $S_i \not\equiv \bot, \forall i \in \{1, .., n\}$. Dans la suite, nous supposerons que Q et les S_i sont donnés par leurs RCF (voir la définition 1.3.2 page 15). Les notations précédentes sont utilisées dans tout le chapitre. Nous supposerons que tous les raisonnements dont on a besoin se font avec des descriptions de concepts dépliées. Suite au chapitre précédent, nous définissons la notion de couverture de Q en utilisant $\mathcal{T}$ comme suit.

Définition 3.1.1 (Couverture d'un concept en utilisant une terminologie)
Une couverture de Q en utilisant $\mathcal{T}$ est une conjonction E de S_i de $\mathcal{S}_\mathcal{T}$ telle que :

$$Q - lcs(Q, E) \not\equiv Q$$

Ainsi, une couverture de Q en utilisant $\mathcal{T}$ est définie comme une conjonction de S_i de $\mathcal{T}$ partageant au moins une information avec Q. On notera qu'une couverture E de Q est toujours consistante avec Q (i.e. $Q \sqcap E \not\equiv \bot$). En effet, quand $\bot$ est un constructeur d'une logique de description à RCF structurellement uniques, ses décompositions non triviales sous la forme d'une conjonction de clauses consistantes ne sont pas possibles (sinon $\mathcal{L}$ n'est pas à RCF structurellement uniques). Dans ces conditions, $\bot$ doit être une clause, ce qui interdit $Q \sqcap E \equiv \bot$.

Pour définir la notion de meilleure couverture, nous avons d'abord besoin de définir le rest, qui est la différence de Q avec une couverture E, et le miss, qui est la différence d'une couverture E avec Q.

Définition 3.1.2 (Rest et miss)
Soit E une couverture de Q en utilisant $\mathcal{T}$.

Le rest de Q par rapport à E, noté $Rest_E(Q)$, est défini comme suit :

$$Rest_E(Q) := Q - lcs(Q, E)$$

Le miss de Q par rapport à E, noté $Miss_E(Q)$, est défini comme suit :

$$Miss_E(Q) := E - lcs(Q, E)$$

Nous rappelons que la différence $C - D$ entre deux descriptions C et D incomparables par rapport à la subsomption se calcule comme la différence $C - lcs(C, D)$ (voir la section 1.3.1 page 14). Cela explique les définitions précédentes des rest et miss.

Nous pouvons maintenant définir la notion de meilleure couverture.

Définition 3.1.3 (Meilleure couverture d'un concept en utilisant une terminologie)
Une $\mathcal{L}$-description de concept E est une meilleure couverture de Q en utilisant $\mathcal{T}$ si et seulement si :

- *E est une couverture de Q en utilisant $\mathcal{T}$,*
- *il n'existe pas de couverture E' de Q en utilisant $\mathcal{T}$ telle que*
  $(|Rest_{E'}(Q)|, |Miss_{E'}(Q)|) < (|Rest_E(Q)|, |Miss_E(Q)|)$
  où $<$ est l'ordre lexicographique et
- *E, en tant qu'ensemble de S_i, est minimal par rapport à l'inclusion.*

avec $|C|$ la taille de C, C étant $\mathcal{L}$-une description, définie comme le nombre d'occurence de noms de concepts et de rôles dans les clauses de la RCF de C, où $\top$ et $\bot$ ne sont pas comptés (notion de taille inspirée de celle définie dans [9]).

Ainsi une meilleure couverture de Q est une couverture qui minimise d'abord le rest puis le miss en taille. La dernière condition sert à éviter les nombreuses possibilités de couvertures dans lesquelles on aurait plusieurs concepts définis S_i strictement équivalents.

On a vu dans le chapitre 2 que l'on pouvait minimiser l'information contenue dans les rest et miss soit en minimisant leur taille, soit en les maximisant par rapport à la subsomption. Le choix qui est fait ici de minimiser par rapport à la taille a été motivé par l'application des meilleures couvertures à la découverte de services (en particulier, comme évoqué au chapitre 2, une grande taille de miss implique un dialogue plus long entre le système et l'utilisateur, ce que l'on veut éviter). Pour les langages ayant la propriété de subsomption structurelle, il est facile de montrer que si une description C subsume une description D, alors la taille de C est plus petite que la taille de D (la réciproque étant fausse). Donc minimiser par rapport à la taille est un peu plus restrictif que maximiser par rapport à la subsomption.

Exemple 7

Soient la $\mathcal{FL}_0$-terminologie $\mathcal{T}$ et la $\mathcal{FL}_0$-description Q suivantes (en forme dépliée) :
- $S_1 \equiv P_1 \sqcap \forall R_1.P_1 \sqcap \forall R_1R_2.P_2 \sqcap \forall R_2R_1.P_2$
- $S_2 \equiv P_1 \sqcap \forall R_1R_2.P_2 \sqcap \forall R_2R_1.P_2 \sqcap P_3 \sqcap \forall R_1R_1.P_4$
- $S_3 \equiv \forall R_1.P_1 \sqcap \forall R_2R_1.P_2 \sqcap \forall R_1R_2.P_2 \sqcap \forall R_2R_1.P_3 \sqcap P_3 \sqcap \forall R_1R_1.P_4 \sqcap P_4$
- $Q \equiv \forall R_1.P_1 \sqcap \forall R_1R_2.P_2 \sqcap \forall R_2.P_2 \sqcap P_3 \sqcap \forall R_1R_1.P_4$

On a donc $\mathcal{S}_\mathcal{T} = \{S_1, S_2, S_3\}$. Une couverture possible est la conjonction $S_1 \sqcap S_2$. Avec un rest valant $\forall R_2.P_2$ de taille 2 et un miss valant $P_1 \sqcap \forall R_2R_1.P_2$ de taille 4, c'est la meilleure couverture de Q en utilisant $\mathcal{T}$.

Vérifions-le en détaillant l'algorithme naïf qui consiste à calculer les rest et miss de toutes les conjonctions possibles de S_i. Calculons d'abord les lcs entre les S_i et Q :
- $lcs(Q, S_1) \equiv \forall R_1.P_1 \sqcap \forall R_1R_2.P_2$
- $lcs(Q, S_2) \equiv \forall R_1R_2.P_2 \sqcap \forall R_1R_1.P_4 \sqcap P_3$
- $lcs(Q, S_3) \equiv \forall R_1.P_1 \sqcap \forall R_1R_2.P_2 \sqcap \forall R_1R_1.P_4 \sqcap P_3$

A partir des lcs on peut calculer les rest et miss. Ils sont regroupés dans le tableau suivant où ils sont ordonnés selon leur taille.

E	$Rest_E(Q)$	$\lvert Rest_E(Q)\rvert$	$Miss_E(Q)$	$\lvert Miss_E(Q)\rvert$
S_1	$\forall R_2.P_2 \sqcap P_3 \sqcap \forall R_1R_1.P_4$	6	$P_1 \sqcap \forall R_2R_1.P_2$	4
S_2	$\forall R_1.P_1 \sqcap \forall R_2.P_2$	4	$P_1 \sqcap \forall R_2R_1.P_2$	4
S_3	$\forall R_2.P_2$	2	$\forall R_2R_1.P_2 \sqcap \forall R_2R_1.P_3 \sqcap P_4$	7
$S_1 \sqcap S_2$	$\forall R_2.P_2$	2	$P_1 \sqcap \forall R_2R_1.P_2$	4
$S_1 \sqcap S_3$	$\forall R_2.P_2$	2	$P_1 \sqcap \forall R_2R_1.P_2 \sqcap \forall R_2R_1.P3 \sqcap P_4$	8
$S_2 \sqcap S_3$	$\forall R_2.P_2$	2	$P_1 \sqcap \forall R_2R_1.P_2 \sqcap \forall R_2R_1.P3 \sqcap P_4$	8
$S_1 \sqcap S_2 \sqcap S_3$	$\forall R_2.P_2$	2	$P_1 \sqcap \forall R_2R_1.P_2 \sqcap \forall R_2R_1.P3 \sqcap P_4$	8

$S_1 \sqcap S_2$ est bien la seule conjonction de S_i de rest puis de miss de taille minimale.

∘

Le problème du calcul de toutes les meilleures couvertures d'une $\mathcal{L}$-description Q en utilisant une terminologie $\mathcal{T}$ est noté $\mathcal{BCOV}(\mathcal{T}, Q)$. Le problème $\mathcal{BCOV}(\mathcal{T}, Q)$ est un problème de réécriture puisqu'il constitue une instance du cadre général de la réécriture (voir la définition 2.1.1 page 22) dans laquelle :

- les langages source $\mathcal{L}_s$ et de terminologie $\mathcal{L}_t$ sont un seul et même langage ayant la propriété de subsomption structurelle,
- le langage destination $\mathcal{L}_d$ est $\{\sqcap, \sqcup\}$,
- la relation ρ est la suivante : $Q \rho E \Leftrightarrow Q - lcs(Q, E) \not\equiv Q$ et
- on cherche à minimiser par rapport à l'ordre $\preceq$ défini ainsi $E' \preceq E$ si et seulement si $(|Rest_{E'}(Q)|, |Miss_{E'}(Q)|) < (|Rest_E(Q)|, |Miss_E(Q)|)$ où $<$ est l'ordre lexicographique, et si ces couples sont égaux alors E', en tant qu'ensemble de S_i, est inclus dans E.

3.1.2 Modélisation du problème $\mathcal{BCOV}(\mathcal{T}, Q)$ avec les hypergraphes

Dans cette section, nous proposons une approche qui permet de calculer les meilleures couvertures autrement qu'avec l'étude de toutes les conjonctions possibles de S_i. Pour cela nous faisons appel aux hypergraphes. Commençons donc par rappeler quelques définitions concernant ce formalisme.

Définition 3.1.4 (Hypergraphe et transversaux [32])
Un hypergraphe $\mathcal{H}$ est un couple (Σ, Γ) où $\Sigma = \{V_1, \ldots, V_n\}$ est un ensembles d'éléments, et $\Gamma = \{\varepsilon_1, \ldots, \varepsilon_m\}$ un ensemble de sous-ensembles de Σ. Les éléments de Σ sont appelés sommets, et les éléments de Γ sont appelés arêtes.

Un ensemble $X \subseteq \Sigma$ est un transversal de $\mathcal{H}$ si pour chaque $\varepsilon \in \Gamma$, $X \cap \varepsilon \neq \emptyset$. Un transversal X est minimal si aucun sous-ensemble strict X' de X n'est un transversal. L'ensemble des transversaux minimaux d'un hypergraphe $\mathcal{H}$ est noté $Tr(\mathcal{H})$.

Nous montrons maintenant que le problème de découverte des meilleures couvertures revient au problème de la recherche des transversaux minimaux de coût minimal d'un hypergraphe (où chaque sous-ensemble de sommets possède un coût).

Définition 3.1.5 (Hypergraphe $\mathcal{H}_{\mathcal{T}Q}$ généré à partir de $\mathcal{T}$ et Q)
Soit une instance $\mathcal{BCOV}(\mathcal{T}, Q)$ du problème de découverte des meilleures couvertures, nous construisons l'hypergraphe $\mathcal{H}_{\mathcal{T}Q} = (\Sigma, \Gamma)$ de la manière suivante :

- *Chaque S_i de $\mathcal{S}_\mathcal{T}$ devient un sommet V_{S_i} de $\mathcal{H}_{\mathcal{T}Q}$. Ainsi $\Sigma = \{V_{S_i}, i \in \{1, .., n\}\}$.*
- *Chaque clause $A_j \in_\equiv Q$, pour $j \in \{1, .., k\}$, devient une arête dans $\mathcal{H}_{\mathcal{T}Q}$, notée ε_{A_j}, avec $\varepsilon_{A_j} = \{V_{S_i} \mid S_i \in \mathcal{S}_\mathcal{T}$ et $A_j \in_\equiv lcs(Q, S_i)\}$.*
  Le symbole $\in_\equiv$ dénote l'appartenance modulo, l'équivalence des clauses, d'une clause à une description donnée par sa RCF.

Pour alléger les démonstrations, nous introduisons les notations suivantes, pour chaque ensemble de sommets $X = \{V_{S_i}\}$, sous-ensemble de Σ :

- on note $E_X \equiv \sqcap_{V_{S_i} \in X} S_i$ le concept obtenu à partir de la conjonction des concepts définis S_i correspondant aux sommets de X
- et réciproquement, pour chaque concept $E \equiv \sqcap_{j \in \{1,..,m\}} S_{i_j}$, on note $X_E = \{V_{S_{i_j}}, j \in \{1, .., m\}\}$ l'ensemble des sommets correspondant aux S_i de E.

Le lemme 3.1.1 donné ci-dessous montre que le rest de taille minimal peut être construit à partir des clauses correspondant aux arêtes vides de l'hypergraphe. Il montre de plus que calculer une couverture de Q en utilisant $\mathcal{T}$ qui minimise le rest revient à calculer un transversal de $\mathcal{H}_{\mathcal{T}Q}$ en considérant seulement les arêtes non vides.

Lemme 3.1.1 (Caractérisation des couvertures de rest minimal en taille)
Soit $\mathcal{L}$ une logique de description ayant la propriété de subsomption structurelle, $\mathcal{T}$ une $\mathcal{L}$-terminologie, avec $\mathcal{S}_\mathcal{T} = \{S_i, i \in \{1,..,n\}\}$, et $Q \equiv \bigsqcap_{j=1}^{k} A_j$ une $\mathcal{L}$-description décrite par sa RCF $\{A_j, j \in \{1,..,k\}\}$. Soit $\mathcal{H}_{\mathcal{T}Q} = (\Sigma, \Gamma)$ l'hypergraphe construit à partir de $\mathcal{T}$ et Q, et $\widehat{\mathcal{H}}_{\mathcal{T}Q} = (\Sigma, \Gamma')$ l'hypergraphe construit en enlevant à $\mathcal{H}_{\mathcal{T}Q}$ les arêtes vides.

• *Quelle que soit une conjonction E de S_i, E est une couverture qui minimise le rest si $Rest_E(Q) \equiv Rest_{min}$ avec $Rest_{min} \equiv \bigsqcap_{\varepsilon_{A_j}=\emptyset} A_j$ (i.e. la conjonction des clauses de Q correspondant aux arêtes vides).*

• *Une description $E \equiv S_{i_1} \sqcap \ldots \sqcap S_{i_m}$, avec $1 \leq m \leq n$ et $S_{i_j} \in \mathcal{S}_\mathcal{T}$ pour $1 \leq j \leq m$, est une couverture de Q en utilisant $\mathcal{T}$ qui minimise $|Rest_E(Q)|$ si et seulement si X_E est un transversal de $\widehat{\mathcal{H}}_{\mathcal{T}Q}$.*

Démonstration
Nous montrons d'abord le premier point : les clauses A_j correspondant aux arêtes vides forment la RCF du rest de taille minimale. En effet, on a, pour $A_j \in_\equiv Q$:

$$\begin{array}{rll} \varepsilon_{A_j} = \emptyset & \Leftrightarrow & \forall S_i \in \mathcal{S}_\mathcal{T},\ V_{S_i} \notin \varepsilon_{A_j} \\ & \Leftrightarrow & \forall S_i \in \mathcal{S}_\mathcal{T},\ A_j \not\in_\equiv lcs(Q, S_i) \\ & \overset{lemme\ A.1.3}{\Leftrightarrow} & A_j \not\in_\equiv lcs(Q, \bigsqcap_{S_i \in \mathcal{S}_\mathcal{T}} S_i) \\ & \overset{A_j \in_\equiv Q}{\Leftrightarrow} & A_j \in_\equiv Rest_{S_1 \sqcap \ldots \sqcap S_n}(Q) \qquad (1) \end{array}$$

Donc l'ensemble des clauses A_j de Q correspondant à une arête vide de l'hypergraphe est exactement l'ensemble des clauses du rest de Q par rapport à la conjonction de tous les S_i de $\mathcal{T}$.
De plus, pour une clause A et une conjonction E de S_i de $\mathcal{T}$, on a :

$$\begin{array}{rll} A \in_\equiv Rest_E(Q) & \Leftrightarrow & A \in_\equiv Q - lcs(Q, E) \\ & \Leftrightarrow & A \in_\equiv Q \setminus_\equiv lcs(Q, E) \\ & \Leftrightarrow & A \in_\equiv Q \text{ et } A \not\in_\equiv lcs(Q, E) \\ & \overset{lemme\ A.1.3}{\Leftrightarrow} & A \in_\equiv Q \text{ et } \forall S_i \in E,\ A \not\in_\equiv lcs(Q, S_i) \qquad (2) \end{array}$$

Donc une clause du rest de Q par rapport à E est une clause de Q qui n'est clause d'aucun lcs de Q et d'un S_i de E.
Ainsi on a pour toute conjonction E de S_i :

$$\begin{array}{rll} \varepsilon_{A_j} = \emptyset & \overset{(1)}{\Leftrightarrow} & A_j \in_\equiv Rest_{S_1 \sqcap \ldots \sqcap S_n}(Q) \\ & \overset{(2)}{\Leftrightarrow} & A_j \in_\equiv Q \text{ et } \forall S_i \in \mathcal{S}_\mathcal{T},\ A_j \not\in_\equiv lcs(Q, S_i) \\ & \overset{E \subseteq \mathcal{S}_\mathcal{T}}{\Rightarrow} & A_j \in_\equiv Q \text{ et } \forall S_i \in E,\ A_j \not\in_\equiv lcs(Q, S_i) \\ & \overset{lemme\ A.1.3}{\Leftrightarrow} & A_j \in_\equiv Rest_E(Q) \end{array}$$

On arrive à la conclusion que :

$$\forall E \subseteq \mathcal{S}_\mathcal{T},\ \{A_j \in_\equiv Q \mid \varepsilon_{A_j} = \emptyset\} \subseteq Rest_E(Q) \qquad (3)$$

En résumé, on a montré que toutes les clauses de Q correspondant à des arêtes vides étaient des clauses du rest de toutes les conjonctions de S_i possibles. Comme, d'après (1), l'ensemble de ces clauses est aussi celui du rest de la conjonction de tous les S_i, alors la conjonction de tous les S_i a un rest minimal en taille puisque tous les autres rest ont soit exactement les mêmes clauses, soit au moins une de plus. On a ainsi démontré le premier point.

Nous montrons maintenant le deuxième point : une conjonction E de S_i est de taille minimale si et seulement si l'ensemble des sommets correspondant est un transversal de $\widehat{\mathcal{H}}_{\mathcal{T}Q}$. D'après le premier point, pour $E = \bigsqcap_{j\in\{1,..,m\}} S_{i_j}$ une conjonction de S_i, on a la première équivalence parmi celles qui suivent :

$$
\begin{array}{lcl}
|Rest_E(Q)| & \Leftrightarrow & Rest_E(Q) = \{A_j \in_{\equiv} Q \mid \varepsilon_{A_j} = \emptyset\} \\
\text{est minimale} & & \\
 & \overset{(3)}{\Leftrightarrow} & Rest_E(Q) \subseteq \{A_j \in_{\equiv} Q \mid \varepsilon_{A_j} = \emptyset\} \\
 & \Leftrightarrow & \forall A_j \in_{\equiv} Q,\ (\varepsilon_{A_j} \neq \emptyset \rightarrow A_j \notin_{\equiv} Rest_E(Q)) \\
 & \overset{Rest_E(Q)=(Q\setminus_{\equiv} lcs(Q,E))}{\Leftrightarrow} & \forall \varepsilon_{A_j} \in \widehat{\mathcal{H}}_{\mathcal{T}Q},\ A_j \notin_{\equiv} (Q \setminus_{\equiv} lcs(Q,E)) \\
 & \overset{A_j\in_{\equiv}Q}{\Leftrightarrow} & \forall \varepsilon_{A_j} \in \widehat{\mathcal{H}}_{\mathcal{T}Q}, A_j \in_{\equiv} lcs(Q,E) \\
 & \overset{lemme\ A.1.3}{\Leftrightarrow} & \forall \varepsilon_{A_j} \in \widehat{\mathcal{H}}_{\mathcal{T}Q}, \exists S_{i_j} \text{ avec } j \in \{1,..,m\} \mid \\
 & & \qquad A_j \in_{\equiv} lcs(Q, S_{i_j}) \\
 & \overset{def.\ 3.1.5}{\Leftrightarrow} & \forall \varepsilon_{A_j} \in \widehat{\mathcal{H}}_{\mathcal{T}Q}, \exists V_{S_{i_j}} \in X_E | V_{S_{i_j}} \in \varepsilon_{A_j} \\
 & \Leftrightarrow & X_E \text{ est un transversal de } \widehat{\mathcal{H}}_{\mathcal{T}Q} \\
 & & (\text{puisque } X_E \text{ intersecte chaque arête de } \widehat{\mathcal{H}}_{\mathcal{T}Q})
\end{array}
$$

□

Ayant déterminé les couvertures qui minimisent le rest, il reste à déterminer celles qui minimisent le miss afin de trouver les meilleures couvertures. Pour exprimer la minimisation du miss dans le contexte des hypergraphes, nous introduisons la notion de coût.

Définition 3.1.6 (Coût d'un ensemble de sommets)
Soit $\mathcal{L}$ une logique de description à subsomption structurelle, $\mathcal{T}$ une $\mathcal{L}$-terminologie, et Q une $\mathcal{L}$-description de concept. Soit $\mathcal{BCOV}(\mathcal{T}, Q)$ le problème de la découverte des meilleures couvertures pour $\mathcal{T}$ et Q, et $\widehat{\mathcal{H}}_{\mathcal{T}Q} = (\Sigma, \Gamma')$ son hypergraphe associé. Le coût de l'ensemble d'un ensemble de sommets X est défini comme suit :

$$cout(X) = |Miss_{E_X}(Q)|$$

Ainsi, $\mathcal{BCOV}(\mathcal{T}, Q)$ peut être réduit au problème du calcul des transversaux de coût minimal de l'hypergraphe $\widehat{\mathcal{H}}_{\mathcal{T}Q}$. Clairement, il apparaît que nous pouvons limiter les recherches aux transversaux minimaux de coût minimal, c'est-à-dire aux transversaux de coût minimal qui sont en plus minimaux par rapport à l'inclusion. En effet, le coût est une fonction monotone par rapport à l'inclusion puisque $X \subseteq Y \Rightarrow cout(X) \leq cout(Y)$. De plus le calcul des transversaux minimaux implique que des ensembles de sommets contenant des sommets relatifs à des S_i équivalents ne seront pas retenus comme solution. Ceci assure la vérification de la troisième et dernière condition définissant les meilleures couvertures (voir la définition 3.1.3 page 28).

En résumé, résoudre $\mathcal{BCOV}(\mathcal{T}, Q)$ revient à résoudre la recherche des transversaux minimaux de coût minimal de l'hypergraphe $\widehat{\mathcal{H}}_{\mathcal{T}Q}$. Comme on le verra dans la section 3.2, on pourra donc réutiliser certains résultats déjà existants du domaine des hypergraphes et des transversaux minimaux.

Exemple 9
Appliquons la modélisation en hypergraphes à l'exemple 7. On obtient l'hypergraphe H_{TQ} avec les 5 arêtes suivantes :

- $\varepsilon_1 = \varepsilon_{\forall R_1.P_1}$,
- $\varepsilon_2 = \varepsilon_{\forall R_1R_2.P_2}$,

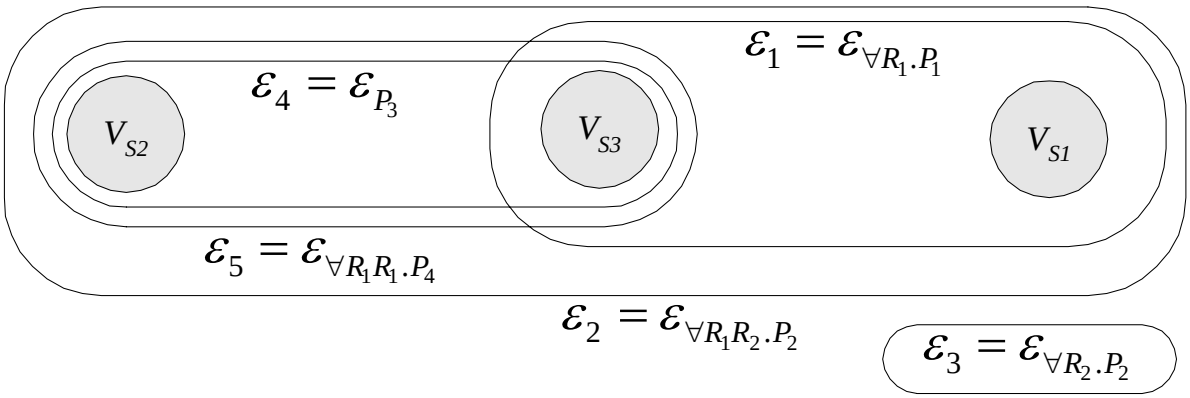


FIG. 3.1 – L'hypergraphe $H_{\mathcal{T}Q}$ construit à partir de Q et de $\mathcal{T}$ dans l'exemple 7.

– $\varepsilon_3 = \varepsilon_{\forall R_2.P_2}$,
– $\varepsilon_4 = \varepsilon_{P_3}$ et
– $\varepsilon_5 = \varepsilon_{\forall R_1 R_1.P_4}$.

En regardant les clauses des lcs de Q avec chaque S_i, on en déduit que les arètes de l'hypergraphes sont remplies ainsi :
– comme $lcs(Q, S_1) \equiv \forall R_1.P_1 \sqcap \forall R_1 R_2.P_2$, on a $V_{S_1} \in \varepsilon_1$ et $V_{S_1} \in \varepsilon_2$,
– comme $lcs(Q, S_2) \equiv \forall R_1 R_2.P_2 \sqcap \forall R_1 R_1.P_4 \sqcap P_3$, on a $V_{S_2} \in \varepsilon_2$, $V_{S_2} \in \varepsilon_4$ et $V_{S_2} \in \varepsilon_5$ et
– comme $lcs(Q, S_3) \equiv \forall R_1.P_1 \sqcap \forall R_1 R_2.P_2 \sqcap \forall R_1 R_1.P_4 \sqcap P_3$, on a $V_{S_3} \in \varepsilon_1$, $V_{S_3} \in \varepsilon_2$, $V_{S_3} \in \varepsilon_4$ et $V_{S_3} \in \varepsilon_5$.

La figure 3.1 montre l'hypergraphe $H_{\mathcal{T}Q}$ qui en découle. On peut voir que les transversaux de $\widehat{H}_{\mathcal{T}Q}$ sont les ensembles suivants : $\{V_{S_3}\}$, $\{V_{S_1}, V_{S_3}\}$, $\{V_{S_2}, V_{S_3}\}$, $\{V_{S_1}, V_{S_2}\}$ et $\{V_{S_1}, V_{S_2}, V_{S_3}\}$. On peut ainsi vérifier, en regardant le tableau de l'exemple 7, que chaque transversal de $\widehat{H}_{\mathcal{T}Q}$ est bien une combinaison dont le rest est de taille minimale.
Pour obtenir les meilleures couvertures, il reste à isoler parmi les transversaux calculés ceux qui sont de coût minimal. Comme évoqué précédemment, on peut se limiter à les rechercher parmi les transversaux minimaux, ici $\{V_{S_3}\}$ et $\{V_{S_1}, V_{S_2}\}$. Après le calcul de leur coût, on trouve que $\{V_{S_1}, V_{S_2}\}$ est la solution, c'est-à-dire que $S_1 \sqcap S_2$ est l'unique meilleure couverture de Q selon $\mathcal{T}$.

◦

3.2 Calcul des meilleures couvertures

Nous présentons maintenant l'algorithme *computeBCov* pour le calcul des transversaux minimaux de coût minimal de l'hypergraphe $\widehat{\mathcal{H}}_{\mathcal{T}Q}$ construit à partir d'une terminologie $\mathcal{T}$ et d'une requête Q. Nous partons d'un algorithme classique de calcul de transversaux minimaux (section 3.2.1). Nous l'améliorons avec le théorème des persistants (section 3.2.2). Puis nous l'adaptons pour découvrir, parmi les transversaux minimaux de $\widehat{\mathcal{H}}_{\mathcal{T}Q}$, ceux qui ont un coût minimal (section 3.2.3). Cette adaptation est basée sur une technique d'optimisation combinatoire dite de séparation et évaluation (ou Branch and Bound, "BnB", en Anglais). Enfin, nous détaillons l'exécution de l'algorithme obtenu sur l'exemple présenté précédemment.

3.2.1 Etat de l'art du calcul des transversaux minimaux dans un hypergraphe

La recherche des transversaux minimaux d'un hypergraphe est un problème central dans de nombreux domaines de l'informatique (voir [32, 33] pour un panorama détaillé). Malgré les nombreuses études sur le sujet, sa complexité précise reste à trouver. Il est clair que c'est un problème exponentiel (en espace et donc en temps) puisque certains hypergraphes ont un nombre de transversaux minimaux exponentiel en fonction de la taille de l'hypergraphe (voir par exemple les hypergraphes H_1 et H_2 définis dans l'annexe C page 177). La principale question qui se pose est de savoir s'il existe un algorithme "output-polynomial", c'est-à-dire polynomial en fonction des tailles de l'entrée (i. e. la taille de l'hypergraphe, son nombre de sommets et d'arêtes) et de la sortie (i. e. la cardinalité de l'ensemble des transversaux minimaux). Ce problème est encore à notre connaissance un problème ouvert.

Dans [37], il est montré que la génération des transversaux minimaux d'un hypergraphe peut être effectuée en temps incrémental sub-exponentiel (presque polynomial) $\mathcal{O}(k^{log(k)})$, où k est la taille combinée de l'entrée et de la sortie. C'est, à notre connaissance la meilleure borne théorique connue pour ce problème. D'autres travaux importants sur la complexité de ce problème sont présents dans [32, 34] : dans ces travaux, des classes d'hypergraphes sont identifiées pour lesquelles le problème est polynômial en fonction des tailles d'entrée et de sortie.

D'un point de vue pratique, plusieurs approches algorithmiques existent. Il y a d'abord l'algorithme de [32] et [61], lui-même fondé sur une propriété donnée dans [16]. C'est l'algorithme classique pour le calcul des transversaux minimaux d'un hypergraphe. La propriété sur laquelle il est fondé permet de calculer les transversaux minimaux d'un hypergraphe $\mathcal{H}$ à partir des transversaux minimaux de $\mathcal{H}'$, avec $\mathcal{H}'$ égal à $\mathcal{H}$ privé d'une arête ε. Ainsi, en partant de $\mathcal{H}$ privé de toutes ses arêtes, on obtient $Tr(\mathcal{H})$ en ajoutant itérativement toutes ses arêtes et en calculant à chaque itération les nouveaux transversaux minimaux à partir de ceux obtenus à l'itération précédente. Ainsi, à l'itération j, on a calculé $Tr({\mathcal{H}'}_{j-1})$ et grâce à cela, on peut calculer $Tr({\mathcal{H}'}_j)$, où ${\mathcal{H}'}_j$ est égal à ${\mathcal{H}'}_{j-1}$ plus l'arête ε_j. Le calcul se fait en deux étapes :

- étape 1 : génération d'un ensemble de transversaux candidats par calcul de toutes les unions possibles entre un transversal minimal $X \in Tr({\mathcal{H}'}_{j-1})$ et un sommet $V_i \in \varepsilon_j$.
- étape 2 : suppression des transversaux candidats qui ne sont pas minimaux.

L'espace à explorer est donc un arbre ayant l'ensemble vide pour racine (on n'a aucun transversal minimal au départ) que l'on parcourt en largeur, les feuilles étant les transversaux minimaux recherchés (ceux de l'hypergraphe entier, avec toutes ses arêtes). Cet algorithme est celui dont on s'est inspiré pour notre problème. Il est rappelé ci-dessous (voir l'algorithme 1).

Algorithme 1 Algorithme classique de calcul des transversaux minimaux d'un hypergraphe

Entrée(s) : Un hypergraphe simple $\mathcal{H} = (\Sigma, \Gamma)$
avec $\Sigma = \{V_i\}$ et $\Gamma = \{\varepsilon_j\}$.
Sortie(s) : $Tr(\mathcal{H})$

1: $Tr := \emptyset$;
2: **pour chaque** arête $\varepsilon_j \in \Gamma$ **faire**
3: *– Etape 1 : génération des candidats –*
$Tr_1 := \{X \cup \{V_i\} \mid X \in Tr \text{ et } V_i \in \varepsilon_j\}$;
4: *– Etape 2 : suppression des candidats non minimaux –*
$Tr_2 := \{X \in Tr_1 \mid X \text{ est minimal par rapport à } \subseteq\}$;
5: $Tr := Tr_2$;
6: **fin pour**
7: Renvoyer Tr ;

Un algorithme basé sur la même propriété mais permettant un parcours en profondeur est proposé dans [47]. Dans cette approche, on n'a pas besoin de calculer tous les transversaux minimaux d'un hypergraphe, pour ensuite pouvoir calculer les transversaux minimaux du même hypergraphe avec une arête supplémentaire. Le gain en termes de place mémoire durant l'exécution est donc important. De plus, les transversaux minimaux ne sont plus donnés en fin de processus, mais égrénés tout au long du calcul, permettant ainsi d'avoir les premières solutions rapidement.

Enfin, deux autres approches ont été proposées. Dans [59] est utilisée une stratégie de calcul par niveaux dans le treillis des parties. Cette technique apparaît comme efficace si la cardinalité moyenne des transversaux minimaux est réduite. Dans les autres cas, elle est très coûteuse en temps de calcul. Dans [78], c'est une énumération des sommets, basée sur l'ordre lexicographique, qui permet de parcourir en profondeur le treillis des parties de l'ensemble des sommets de l'hypergraphe et ainsi de découvrir les transversaux minimaux. Par rapport à l'approche précédente, les résultats en pratique sont meilleurs car le parcours en profondeur permet une utilisation plus réduite de la mémoire.

Dans cette thèse, nous avons utilisé l'algorithme de base implémentant un parcours en largeur optimisé par une propriété nommée théorème des persistants que l'on explique dans la suite

3.2.2 Théorème des persistants

Dans cette section, nous présentons l'optimisation de l'algorithme classique évoquée précédemment et nommée théorème des persistants. Nous avons vu que l'algorithme classique était itératif et que chaque itération était composée d'une étape de génération de transversaux candidats suivie d'une étape de suppression des candidats non minimaux. Le principe de l'amélioration proposée ici est d'éviter la deuxième étape en générant directement les transversaux minimaux. Le théorème 3.2.1 formalise cette idée. L'algorithme qui en découle est l'algorithme 2.

Supposons que l'on étudie l'itération k : à partir des transversaux de $\mathcal{H}$ générés à l'itération $k-1$ et à partir de l'arête ε nous devons générer les transversaux minimaux de $\mathcal{H}' = \mathcal{H} \cup \varepsilon$. L'idée du théorème 3.2.1 est de classer les transversaux minimaux obtenus à l'itération $k-1$ en plusieurs catégories. D'abord, il y a les "persistants", c'est à dire ceux qui ont une intersection non vide avec ε : il est clair que ces transversaux, qui sont minimaux à l'itération $k-1$, le seront aussi à l'itération k puisqu'ils intersectent ε. C'est pour cette raison qu'on les nomme "persistants". Parmi les persistants, nous distinguons les "n-persistants" qui ont au moins deux sommets en commun avec l'arête ε, et les "1-persistants" qui n'ont qu'un seul sommet en commun avec ε. Nous déduisons de cela que les éventuels nouveaux transversaux minimaux de l'itération k seront construits comme union d'un sommet de ε et d'un des autres transversaux minimaux de l'itération $k-1$. Ces autres transversaux minimaux ont une intersection vide avec ε. On les nomme "non-persistants". Le point a. du théorème 3.2.1, ou "théorème des persistants", résume ce qui vient d'être dit.

Le point b. du théorème des persistants donne quant à lui une condition nécessaire et suffisante pour caractériser les non-persistants X_i dont l'union avec un sommet V_j de ε génèrera un transversal non minimal à l'itération k. Cette condition nécessaire et suffisante est l'existence d'un 1-persistant X_k dont l'intersection avec ε est justement le singleton $\{V_j\}$, et l'inclusion stricte de $X_k \setminus \{V_j\}$ dans X_i.

Ce point b. est le cœur du théorème des persistants, et constitue une contribution algorithmique nouvelle de cette thèse.

Théorème 3.2.1 (Théorème des persistants)
Soit $\mathcal{H}$ un hypergraphe et $Tr(\mathcal{H}) = \{X_i \mid i \in \{1,..,m\}\}$ l'ensemble de ses transversaux minimaux. Soit $\varepsilon = \{V_j \mid j \in \{1,..,n\}\}$ une arête supplémentaire construite avec les sommets de $\mathcal{H}$. Soit $\mathcal{H}' = \mathcal{H} \cup \varepsilon$.
Pour tout X_i transversal minimal de $\mathcal{H}$, on a :

a. *Cas où X_i est un persistant : i.e. $X_i \cap \varepsilon \neq \emptyset$:*
Pour tout sommet V_j de ε :
– $(V_j \notin X_i \cap \varepsilon) \Rightarrow (X_i \cup \{V_j\}$ *est un transversal de $\mathcal{H}'$ qui n'est pas minimal*$)$
– $(V_j \in X_i \cap \varepsilon) \Rightarrow (X_i \cup \{V_j\} = X_i$ *est un transversal minimal de $\mathcal{H}'$*$)$

b. *Cas où X_i est un non-persistant : i.e. $X_i \cap \varepsilon = \emptyset$:*
Pour tout sommet V_j de ε :
$(X_i \cup \{V_j\}$ *est un transversal de $\mathcal{H}'$ qui n'est pas minimal*$) \Leftrightarrow (\exists X_k \in Tr(\mathcal{H}) \mid X_k \cap \varepsilon = \{V_j\}$ *et* $X_k \setminus \{V_j\} \subset X_i)$

Démonstration
a. : Immédiat.
b. : Nous rappelons que (1) $X_i \cap \varepsilon = \emptyset$, (2) $X_i \in Tr(\mathcal{H})$, (3) $\mathcal{H}' = \mathcal{H} \cup \varepsilon$ et (4) $V_j \in \varepsilon$. On rappelle aussi que $Tr(\mathcal{H})$ dénote l'ensemble de tous les transversaux minimaux de l'hypergraphe $\mathcal{H}$ alors que $tr(\mathcal{H})$ dénote l'ensemble de tous les transversaux (pas uniquement minimaux) de l'hypergraphe $\mathcal{H}$. Soit $(*) = X_i \cup \{V_j\}$ est un transversal non minimal de $\mathcal{H}'$.
Soit $(**) = \exists X_k \in Tr(\mathcal{H}) \mid X_k \cap \varepsilon = \{V_j\}$ et $X_k \setminus \{V_j\} \subset X_i$.
Montrons l'équivalence entre $(*)$ et $(**)$.

$$(*) \overset{(1,2,3)}{\Leftrightarrow} \exists Y \mid Y \in Tr(\mathcal{H}') \text{ et } Y \subset X_i \cup \{V_j\} \overset{(1,2,3,4)}{\Leftrightarrow} \exists Y \mid Y \in Tr(\mathcal{H}') \text{ et } V_j \in Y \text{ et } Y \setminus \{V_j\} \subset X_i$$

$$\Leftrightarrow \exists Y \mid \forall \varepsilon' \in \mathcal{H}',\ Y \cap \varepsilon' \neq \emptyset \text{ et } \forall V_y \in Y,\ Y \setminus \{V_y\} \notin tr(\mathcal{H}') \text{ et } V_j \in Y \text{ et } Y \setminus \{V_j\} \subset X_i \Leftrightarrow \exists Y \mid \forall \varepsilon' \in \mathcal{H}',\ Y \cap \varepsilon' \neq \emptyset \text{ et } Y \setminus \{V_j\} \notin tr(\mathcal{H}') \text{ et } \forall V_y \in Y,\ V_y \neq V_j,\ Y \setminus \{V_y\} \notin tr(\mathcal{H}') \text{ et } V_j \in Y \text{ et } Y \setminus \{V_j\} \subset X_i$$

$$\overset{(2,3)}{\Leftrightarrow} \exists Y \mid \forall \varepsilon' \in \mathcal{H}',\ Y \cap \varepsilon' \neq \emptyset \text{ et } Y \setminus \{V_j\} \notin tr(\mathcal{H}) \text{ et } \forall V_y \in Y, V_y \neq V_j, Y \setminus \{V_y\} \notin tr(\mathcal{H}') \text{ et } V_j \in Y \text{ et } Y \setminus \{V_j\} \subset X_i \overset{(3,4)}{\Leftrightarrow} \exists Y \mid \forall \varepsilon' \in \mathcal{H}',\ Y \cap \varepsilon' \neq \emptyset \text{ et } Y \setminus \{V_j\} \notin tr(\mathcal{H}) \text{ et } \forall V_y \in Y, V_y \neq V_j, Y \setminus \{V_y\} \notin tr(\mathcal{H}) \text{ et } V_j \in Y \text{ et } Y \setminus \{V_j\} \subset X_i$$

$$\overset{(3,4)}{\Leftrightarrow} \exists Y \mid \forall \varepsilon' \in \mathcal{H},\ Y \cap \varepsilon' \neq \emptyset \text{ et } Y \setminus \{V_j\} \notin tr(\mathcal{H}) \text{ et } \forall V_y \in Y,\ V_y \neq V_j,\ Y \setminus \{V_y\} \notin tr(\mathcal{H}) \text{ et } V_j \in Y \text{ et } Y \setminus \{V_j\} \subset X_i \Leftrightarrow \exists Y \mid Y \in Tr(\mathcal{H}) \text{ et } V_j \in Y \text{ et } Y \setminus \{V_j\} \subset X_i$$

$$\overset{(1,4)}{\Leftrightarrow} \exists Y \mid Y \in Tr(\mathcal{H}) \text{ et } Y \cap \varepsilon = V_j \text{ et } Y \setminus \{V_j\} \subset X_i \Leftrightarrow (**)$$

□

L'algorithme 2 implémente le théorème 3.2.1 des persistants sur la base de l'algorithme 1. En dehors du théorème 3.2.1, le seul autre ajout est l'initialisation de l'ensemble courant des transversaux minimaux à un élément contenant tous les sommets des arêtes à un seul sommet, au lieu de l'ensemble vide, ce qui ne change rien au fonctionnement de l'algorithme.

Le théorème des persistants évite l'étape de suppression des candidats non minimaux au prix

Algorithme 2 Algorithme classique de génération des transversaux minimaux d'un hypergraphe amélioré avec le théorème 3.2.1 des persistants.

Entrée(s) : Un hypergraphe simple $\mathcal{H} = (\Sigma, \Gamma)$.
Sortie(s) : $Tr(\mathcal{H})$

1: $Tr := \{\{V\} \mid V \in \Sigma \text{ et } \exists \varepsilon \in \Gamma \mid \varepsilon = \{V\}\}$;
2: **pour chaque** arête $\varepsilon \in \Gamma \mid |\varepsilon| \geq 2$ **faire**
3: $Tr_{np} := \emptyset$;
4: $Tr_{1p} := \emptyset$;
5: $Tr_{notp} := \emptyset$;
6: **pour chaque** $X \in Tr$ **faire**
7: $I := \varepsilon \cap X$;
8: **si** $|I| \geq 2$ **alors**
9: $Tr_{np} := Tr_{np} \cup \{X\}$;
10: **sinon si** $|I| = 1$ **alors**
11: $Tr_{1p} := Tr_{1p} \cup \{(X, I)\}$;
12: **sinon**
13: $Tr_{notp} := Tr_{notp} \cup \{(X, \varepsilon)\}$;
14: **fin si**
15: **fin pour**
16: **pour chaque** $(X, I) \in Tr_{1p}$ **faire**
17: **pour chaque** $(X', \varepsilon') \in Tr_{notp} | (X \setminus I) \subset X'$ **faire**
18: $\varepsilon' := \varepsilon' \setminus I$;
19: **fin pour**
20: **fin pour**
21: $Tr := Tr_{np}$;
22: **pour chaque** $(X, I) \in Tr_{1p}$ **faire**
23: $Tr := Tr \cup \{X\}$;
24: **fin pour**
25: **pour chaque** $(X', \varepsilon') \in Tr_{notp}$ **faire**
26: **pour chaque** vertex $V' \in \varepsilon'$ **faire**
27: $Tr := Tr \cup \{X' \cup \{V'\}\}$;
28: **fin pour**
29: **fin pour**
30: **fin pour**

d'un nombre d'opérations supplémentaires dont le coût total est inférieur, dans la génération des candidats. C'est donc pourquoi on peut considérer ce théorème comme une amélioration de l'algorithme 1. Une étude approfondie prouvant cette conclusion est donnée dans l'annexe C page 177.

3.2.3 L'algorithme *computeBCov*

Nous rappelons que le problème à résoudre est celui du calcul des transversaux minimaux de coût minimal. Nous présentons donc comment nous avons adapté l'algorithme 2 de calcul des transversaux minimaux au problème de calcul des transversaux minimaux de coût minimal pour l'hypergraphe $\widehat{\mathcal{H}}_{\mathcal{T}Q}$. Le résultat est l'algorithme 3, nommé *computeBCov* et donné à la fin de cette section. Son implémentation et les tests qu'il a permis de réaliser sont développés dans le chapitre 7.

En partant de l'algorithme 2, l'approche naïve est de calculer tous les transversaux minimaux, puis de les trier par rapport à leur coût afin de ne conserver que ceux qui ont un coût minimal. Au lieu de cela, nous utilisons la structure itérative de l'algorithme pour supprimer, à chaque itération, les transversaux qui sont minimaux pour cette itération mais dont on est sûr qu'ils ne génèreront aucun transversal minimal de coût minimal pour l'hypergraphe entier.

Pour effectuer ces suppressions, l'idée principale de *computeBCov* est d'utiliser une technique d'optimisation combinatoire nommée "séparation et évaluation" (ou Branch and Bound, "BnB", en Anglais), pour élaguer des branches de l'arbre de recherche et ainsi se diriger plus rapidement vers les solutions recherchées. Au début de *computeBCov* (ligne 3), une heuristique simple est utilisée pour calculer un transversal (pas forcément minimal) de l'hypergraphe entier ayant un coût faible. Un tel transversal est obtenu en ajoutant, pour chaque arête de l'hypergraphe, le coût du sommet de plus petit coût. Le coût total obtenu par l'heuristique est stocké dans la variable $CostEval$. Comme nous avons, pour tout ensemble de sommets $X = \{V_{S_i}\}$:

$$cout(X) = |Miss_{E_X}(Q)| \leq \sum_i |Miss_{S_i}(Q)| = \sum_{V_{S_i} \in X} cout(\{V_{S_i}\})$$

alors la valeur stockée dans $CostEval$ est une borne supérieure du coût d'un transversal minimal de l'hypergraphe entier. Or, l'ajout d'un sommet à un ensemble de sommets fait augmenter son coût (au moins de 0). Ainsi, on est sûr que tous les transversaux minimaux des itérations intermédiaires qui possèdent un coût strictement supérieur au coût de $CostEval$ ne génèreront au final que des transversaux minimaux ayant un coût strictement supérieur à $CostEval$. Comme le coût stocké dans $CostEval$ est celui d'un transversal de l'hypergraphe entier, on peut supprimer ces transversaux minimaux intermédiaires tout de suite (juste après leur génération). Ainsi, à chaque itération, après l'étape de génération des transversaux minimaux (ligne 5), nous éliminons ceux qui ont un coût strictement supérieur à $CostEval$ (lignes 8 et 9). Ensuite, à partir de chaque transversal minimal restant, nous calculons dans une nouvelle évaluation du coût d'un transversal de l'hypergraphe entier construit à partir du transversal examiné (ligne 11). Si l'une de ces nouvelles évaluations est strictement plus petite que $CostEval$ alors $CostEval$ prend sa valeur (lignes 12 et 13).

Pour calculer une nouvelle évaluation du coût d'un transversal de l'hypergraphe entier construit à partir d'un transversal minimal intermédiaire (ligne 11), nous avons deux politiques possibles :

- Politique 1 : utiliser le principe de l'heuristique qui calcule $CostEval$ au début de l'algorithme, i.e. au coût du transversal intermédiaire (stocké dans $RealCost$), on ajoute le coût du sommet de plus petit coût pour chaque arête qui n'a pas été encore examinée.

– Politique 2 : au coût du transversal intermédiaire (stocké dans *RealCost*), on ajoute le coût du sommet de plus petit coût pour chaque arête qui n'a pas été encore examinée et qui a une intersection vide avec le transversal intermédiaire.

C'est la politique 2 qui est utilisée dans l'algorithme 3. En effet, elle apparaît a priori comme plus performante puisque ses évaluations sont toujours au moins aussi bonnes que celles de la politique 1, et donc le BnB converge d'autant plus rapidement vers les solutions. Cependant, elle nécessite quelques calculs d'intersections supplémentaires. Comme il est difficile de se faire une idée sur le gain relatif dû à une meilleure évaluation par rapport au coût supplémentaire dû aux calculs d'intersections, nous avons implémenté les deux politiques (voir la section 7.2 page 119). En pratique, les expérimentations réalisées montrent que la politique 2 est meilleure que la politique 1 (voir le chapitre 7).

A la fin de l'algorithme, chaque transversal minimal de l'hypergraphe entier est traduit en la solution correspondante du problème $\mathcal{BCOV}(\mathcal{T}, Q)$.

Algorithme 3 *computeBCov*, algorithme de calcul des meilleures couvertures d'un concept en utilisant une terminologie, pour un langage ayant la propriété de subsomption structurelle.

Entrée(s) : Une instance $\mathcal{BCOV}(\mathcal{T}, Q)$ du problème de découverte des meilleures couvertures d'un concept pour un langage à subsomption structurelle.

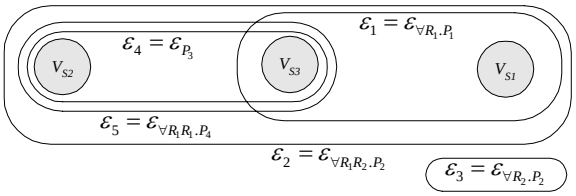
Sortie(s) : L'ensemble des meilleures couvertures de Q selon $\mathcal{T}$.

1: Construction de l'hypergraphe $\widehat{\mathcal{H}}_{\mathcal{T}Q} = (\Sigma, \Gamma')$.
2: $Tr \leftarrow \emptyset$.
3: $CostEval \leftarrow \sum_{\varepsilon \in \Gamma'} \min_{V_{S_i} \in \varepsilon} (|Miss_{S_i}(Q)|)$;
4: **pour chaque** arête $\varepsilon \in \Gamma'$ **faire**
5: $Tr \leftarrow$ {transversaux minimaux générés selon l'algorithme 2}
6: **pour chaque** transversal minimal $X \in Tr$ **faire**
7: $RealCost \leftarrow |Miss_{E_X}(Q)|$;
8: **si** $RealCost > CostEval$ **alors**
9: $Tr \leftarrow Tr \setminus X$;
10: **sinon si** $RealCost < CostEval$ **alors**
11: $Eval \leftarrow RealCost + \sum_{f \in \Gamma' |\ f \cap X = \emptyset} \min_{V_{S_i} \in f} (|Miss_{S_i}(Q)|)$;
12: **si** $Eval < CostEval$ **alors**
13: $CostEval \leftarrow Eval$;
14: **fin si**
15: **fin si**
16: **fin pour**
17: **fin pour**
18: **pour chaque** $X \in Tr$ tel que $|Miss_{E_X}(Q)| = CostEval$ **faire**
19: Retourner le concept E_X.
20: **fin pour**

3.2.4 Exemple d'exécution de *computeBCov*

Nous reprenons maintenant l'exemple 7 page 29 et nous détaillons son exécution avec *computeBCov*. Cette exécution a été effectuée sur le prototype D^2CP qui implémente *computeBCov*

$S_1 \equiv P_1 \sqcap \forall R_1.P_1 \sqcap \forall R_1R_2.P_2 \sqcap \forall R_2R_1.P_2$
$S_2 \equiv P_1 \sqcap \forall R_1R_2.P_2 \sqcap \forall R_2R_1.P_2 \sqcap P_3 \sqcap \forall R_1R_1.P_4$
$S_3 \equiv \forall R_1.P_1 \sqcap \forall R_2R_1.P_2 \sqcap \forall R_1R_2.P_2 \sqcap \forall R_2R_1.P_3 \sqcap P_3 \sqcap \forall R_1R_1.P_4 \sqcap P_4$
$Q \equiv \forall R_1.P_1 \sqcap \forall R_1R_2.P_2 \sqcap \forall R_2.P_2 \sqcap P_3 \sqcap \forall R_1R_1.P_4$



TAB. 3.1 – Exemple d'instance de $\mathcal{BCOV}(\mathcal{T}, Q)$: rappel de l'exemple de la section 7 page 29.

et qui est présenté dans la section 7.2.

Le tableau 3.1 rappelle les principaux composants de cet exemple, c'est-à-dire les S_i de $\mathcal{T}$ et Q dans leur forme dépliée, ainsi que l'hypergraphe $H_{\mathcal{T}Q}$ correspondant.

Détaillons maintenant l'exécution de *computeBCov* sur l'hypergraphe $H_{\mathcal{T}Q}$. L'algorithme *computeBCov* utilisé ici utilise comme on l'a vu l'algorithme 2 pour générer les transversaux minimaux et le Branch and Bound associé à la politique 2 pour ne garder que ceux de coût minimal.

Le tableau 3.1 montre que l'hypergraphe possède en fait trois arêtes différentes non vides et ne contenant strictement aucune autre arête. Donc il y a trois itérations en tout dans *computeBCov* :

- Initialisation :
  - le premier transversal est l'ensemble vide (on n'a encore examiné aucune arête),
  - la première évaluation d'un transversal de l'hypergraphe entier induit un coût de 12 (somme des coûts des sommets de plus petit coût de chaque arête) : donc $CostEval$ est initialisé à 12.
- Itération 1 :
  - L'unique transversal existant est l'ensemble vide.
  - L'arête courante est $\{S_1, S_3\}$.
  - On calcule les intersections entre l'ensemble vide et l'arête courante : le résultat est l'ensemble vide, donc, il n'y a aucun 1- ni n- persistant, par contre il y a un non-persistant (c'est l'ensemble vide). A partir de ce non-persistant et de chaque sommet de l'arête courante, on génère un nouveau transversal minimal. Ainsi 2 transversaux sont générés : $\{S_1\}$ et $\{S_3\}$.
  - Branch and Bound :
    - Le coût de $\{S_3\}$ est 7 (i.e. $|Miss_{S_3}(Q)| = 7$), et à partir de $\{S_3\}$, on peut construire (en prenant le sommet de plus petit coût pour toutes les arêtes non encore examinées ayant une intersection vide avec $\{S_3\}$) un transversal de l'hypergraphe entier ayant un coût de 7. Donc $CostEval := 7$.
    - Le coût de $\{S_1\}$ est 4, et à partir de $\{S_1\}$, un transversal de l'hypergraphe entier peut être construit avec un coût de 8. Donc $CostEval$ reste à 7.
- Itération 2 :
  - L'ensemble des transversaux minimaux calculés à l'itération précédente est $\{\{S_1\}, \{S_3\}\}$.
  - $CostEval$ vaut 7.
  - L'arête courante est $\{S_3, S_2\}$.
  - On calcule l'intersection entre chaque transversal calculé à l'itération précédente et l'arête courante. $\{S_3\}$ est un 1-persistant (parce que $|\{S_3\} \cap \{S_3, S_2\}| = 1$), et $\{S_1\}$ est un non-persistant (parce que $|\{S_1\} \cap \{S_3, S_2\}| = 0$). Comme $\{S_3\} \setminus (\{S_3\} \cap \{S_3, S_2\}) = \emptyset \subset \{S_1\}$, alors nous savons, selon le théorème 3.2.1, que $\{S_1\} \cup (\{S_3\} \cap \{S_3, S_2\}) =$

$\{S_1\} \cup \{S_3\}$ ne sera pas un transversal minimal de l'itération courante. Donc la seule génération d'un transversal minimal est ici $\{S_1\} \cup \{S_2\}$.
  - Branch and Bound :
    - Le coût de $\{S_3\}$ est toujours de 7, donc, comme $CostEval$ vaut aussi 7, nous savons qu'il est inutile d'essayer de trouver à partir de $\{S_3\}$ un nouveau transversal de l'hypergraphe entier ayant un coût strictement plus petit que 7. Malgré tout $\{S_3\}$ peut encore être un transversal minimal de coût minimal de l'hypergraphe entier. C'est pourquoi il n'est pas supprimé.
    - Le coût de $\{S_1, S_2\}$ est 4, et à partir de $\{S_1, S_2\}$ un transversal de l'hypergraphe entier peut être construit avec un coût de 4. Donc $CostEval := 4$.
- Itération 3 :
  - L'ensemble des transversaux minimaux calculés à cette itération est $\{\{S_3\}, \{S_1, S_2\}\}$.
  - $CostEval$ vaut 4.
  - L'arête courante est $\{S_1, S_3, S_2\}$.
  - $\{S_3\}$ est un 1-persistant (parce que $|\{S_3\} \cap \{S_1, S_3, S_2\}| = 1$), et $\{S_1, S_2\}$ est un n-persistant (parce que $|\{S_1, S_2\} \cap \{S_1, S_3, S_2\}| = 2 > 1$). Comme il n'y a pas d'autres transversaux minimaux calculés à l'itération précédente, alors aucun nouveau transversal minimal n'est généré à cette itération. On a donc toujours $\{S_3\}$ et $\{S_1, S_2\}\}$.
  - Branch and Bound :
    - Le coût de $\{S_3\}$ est toujours à 7, et donc, comme $CostEval$ vaut 4, on en déduit que $\{S_3\}$ n'est pas un transversal de coût minimal de l'hypergraphe entier. Il est donc effacé.
    - Le coût de $\{S_1, S_2\}$ est 4. Comme $CostEval$ vaut aussi 4, on ne fait rien de particulier.
- Fin de l'algorithme : il n'y a plus d'arête, donc il reste à chercher parmi les transversaux qui restent ceux qui ont un coût égal à $CostEval$. En effet, $CostEval$ a pu être modifié lors de l'examen du dernier transversal généré et il peut subsister des transversaux de coût plus grand. Ici, seul reste $\{S_1, S_2\}$, donc $\{S_1, S_2\}$ est la seule combinaison de S_i qui couvre au mieux Q.

A titre de vérification, une trace de cette exécution réalisée avec le programme D^2CP est disponible à la figure 7.3 page 121.

3.3 Complexité des meilleures couvertures

Dans cette section, nous étudions la complexité du problème $\mathcal{BCOV}(\mathcal{T}, Q)$ et celle de l'algorithme *computeBCov*.

3.3.1 Complexité du problème $\mathcal{BCOV}(\mathcal{T}, Q)$

Nous montrons ici que $\mathcal{BCOV}(\mathcal{T}, Q)$ est un problème NP-Difficile. Pour cela, on utilise un autre problème, proche, qui est celui du calcul d'un transversal de cardinalité minimale dans un hypergraphe, connu comme étant NP-Difficile[10], pour obtenir une borne inférieure de la complexité de $\mathcal{BCOV}(\mathcal{T}, Q)$. On montre en effet que le problème du calcul d'un transversal de cardinalité minimale dans un hypergraphe peut se réduire en temps polynomial au problème $\mathcal{BCOV}(\mathcal{T}, Q)$. La figure 3.2 montre le lien entre $\mathcal{BCOV}(\mathcal{T}, Q)$ et les problèmes relatifs aux hypergraphes évoqués précédemment.

[10]Le fait que la recherche d'un transversal de cardinalité minimale soit un problème NP-Difficile est une conséquence immédiate du fait que le "HITTING SET problem", ou problème de l'existence d'un transversal de cardi-

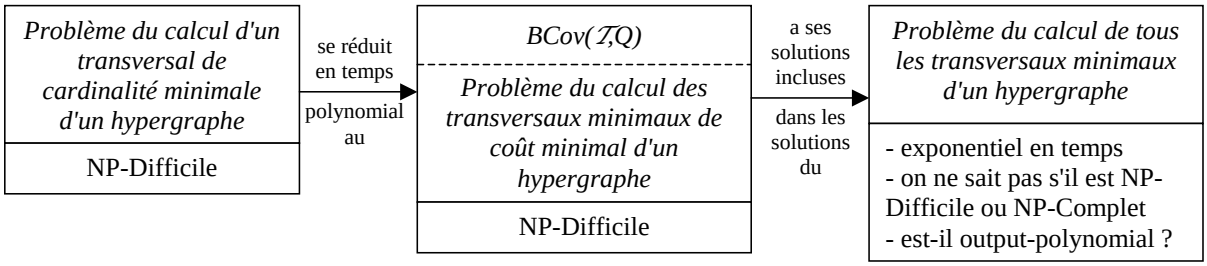


FIG. 3.2 – Le problème $\mathcal{BCOV}(\mathcal{T}, Q)$ et ses liens avec les problèmes de transversaux minimaux d'un hypergraphe.

Théorème 3.3.1 (Complexité de $\mathcal{BCOV}(\mathcal{T}, Q)$)
Soit $\mathcal{L}$ une logique de description à subsomption structurelle, $\mathcal{T}$ une $\mathcal{L}$-terminologie, et Q une $\mathcal{L}$-description de concept.

$\mathcal{BCOV}(\mathcal{T}, Q)$ est un problème NP-Difficile.

Pour démontrer ce théorème, commençons par définir le problème de calcul d'un transversal de cardinalité minimale. Ce problème peut être formulé comme suit : soit $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ un hypergraphe[11] avec $\mathcal{V} = \{v_1, \ldots, v_n\}$ et $\mathcal{E} = \{e_1, \ldots, e_m\}$ (tels que $e_j \subseteq \mathcal{V}$ pour tout $j \in \{1, .., m\}$). Le problème est alors de trouver un transversal de cardinalité minimale[12] de $\mathcal{H}$. Nous montrons maintenant comment on réduit le problème précédent dans $\mathcal{BCOV}(\mathcal{T}, Q)$.

Soit un hypergraphe $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, nous construisons une terminologie $\mathcal{T}_\mathcal{H}$ et une description de concept $Q_\mathcal{H}$ comme suit :

– Chaque arête e_j de $\mathcal{E}$ correspond à une clause A_j de $Q_\mathcal{H}$. $Q_\mathcal{H}$ est donc définie ainsi :
$$Q_\mathcal{H} \equiv \bigsqcap_{j \mid e_j \in \mathcal{E}} A_j$$
– Chaque sommet v_i de $\mathcal{V}$ correspond à un concept S_{v_i} de $\mathcal{T}_\mathcal{H}$ défini comme suit :
$$S_{v_i} \equiv (\bigsqcap_{j \mid v_i \in e_j} A_j) \sqcap B_i, \forall i \in \{1, .., n\}$$

Les A_j et les B_i sont choisis comme des concepts atomiques différents deux à deux. $Q_\mathcal{H}$ et les S_{v_i} sont donc donnés sous forme de RCF. Notons que $Q_\mathcal{H}$ et $\mathcal{T}_\mathcal{H}$ peuvent toujours être construites en temps linéaire en la taille de $\mathcal{H}$ puisque $\mathcal{H}$ n'est parcouru qu'une seule fois.

Le lemme suivant décrit les propriétés de $Q_\mathcal{H}$ et $\mathcal{T}_\mathcal{H}$.

Lemme 3.3.1
Soit $\mathcal{T}_\mathcal{H}$ et $Q_\mathcal{H}$ la terminologie et la description construites à partir de l'hypergraphe $\mathcal{H}$ comme décrit précédemment. On a :

(1) $\widehat{\mathcal{H}}_{\mathcal{T}_\mathcal{H} Q_\mathcal{H}} = \mathcal{H}$, et

(2) $|Miss_E(Q_\mathcal{H})| = |X_E|$ pour toute conjonction E de S_{v_i}.

Démonstration
(1) D'après la définition 3.1.5, nous pouvons construire l'hypergraphe $\mathcal{H}_{\mathcal{T}_\mathcal{H} Q_\mathcal{H}} = (\Sigma_\mathcal{H}, \Gamma_\mathcal{H})$ de la manière suivante :

– Chaque concept défini $S_{v_i} \in \mathcal{T}_\mathcal{H}$, pour $i \in \{1, .., n\}$, devient un sommet $V_{S_{v_i}}$ de $\mathcal{H}_{\mathcal{T}_\mathcal{H} Q_\mathcal{H}}$. Ainsi $\Sigma_\mathcal{H} = \{V_{S_{v_i}}, i \in \{1, .., n\}\}$.

nalité plus petite que k, pour k donné, est NP-complet, voir [38] et [32].

[11] Nous supposons que $\mathcal{H}$ ne contient pas d'arête vide puisque le problème n'est défini que pour ce type d'hypergraphe.

[12] Un transversal de cardinalité minimal est forcément minimal, mais l'inverse n'est pas toujours vrai.

– Chaque clause $A_j \in Q_{\mathcal{H}}$ devient une arête de $\mathcal{H}_{\mathcal{T}_{\mathcal{H}}Q_{\mathcal{H}}}$, notée ε_{A_j}, avec $\varepsilon_{A_j} = \{V_{S_{v_i}} | S_{v_i} \in \mathcal{T}_{\mathcal{H}}$ et $A_j \in_{\equiv} lcs(Q_{\mathcal{H}}, S_{v_i})\}$.

Il est clair qu'il n'y a pas d'arête vide dans $\mathcal{H}_{\mathcal{T}_{\mathcal{H}}Q_{\mathcal{H}}}$, et donc $\mathcal{H}_{\mathcal{T}_{\mathcal{H}}Q_{\mathcal{H}}} = \widehat{\mathcal{H}}_{\mathcal{T}_{\mathcal{H}}Q_{\mathcal{H}}}$. D'après la construction des S_{v_i} et de $Q_{\mathcal{H}}$, et de $\mathcal{H}_{\mathcal{T}_{\mathcal{H}}Q_{\mathcal{H}}}$, on a :

$$(v_i \in e_j) \Leftrightarrow (A_j \in Q_{\mathcal{H}} \text{ et } A_j \in S_{v_i}) \Leftrightarrow (A_j \in_{\equiv} lcs(Q_{\mathcal{H}}, S_{v_i})) \Leftrightarrow (V_{S_{v_i}} \in \varepsilon_{A_j})$$

Dès lors il est clair que $\widehat{\mathcal{H}}_{\mathcal{T}_{\mathcal{H}}Q_{\mathcal{H}}} = \mathcal{H}$.

(2) Soit E une conjonction de S_{v_i}, i.e. $E \equiv \bigsqcap_{h\in\{1,..,p\}} S_{v_{i_h}}$, pour $1 \leq p \leq n$. Par construction de $Q_{\mathcal{H}}$ et $\mathcal{T}_{\mathcal{H}}$, on a :

$$Miss_E(Q_{\mathcal{H}}) \equiv \bigsqcap_{h\in\{1,..,p\}} B_{i_h}$$

Puisque les B_i sont des concepts atomiques, on a donc :

$$|Miss_E(Q_{\mathcal{H}})| = p$$

Comme $\widehat{\mathcal{H}}_{\mathcal{T}_{\mathcal{H}}Q_{\mathcal{H}}} = \mathcal{H}$ (d'après (1)), nous pouvons utiliser la notation X_E. D'où :

$$|Miss_E(Q_{\mathcal{H}})| = p = |X_E|$$

□

Le lemme suivant montre l'équivalence entre trouver un transversal de cardinalité minimale de $\mathcal{H}$ et trouver une meilleure couverture de $Q_{\mathcal{H}}$ en utilisant $\mathcal{T}_{\mathcal{H}}$.

Lemme 3.3.2
Soit $\mathcal{T}_{\mathcal{H}}$ et $Q_{\mathcal{H}}$ la terminologie et la description construites à partir de $\mathcal{H}$. X est un transversal de cardinalité minimale de $\mathcal{H}$ si et seulement si E_X est une meilleure couverture de $Q_{\mathcal{H}}$ en utilisant $\mathcal{T}_{\mathcal{H}}$.

Démonstration

X est un transversal de cardinalité minimale de $\mathcal{H}$	$\Leftrightarrow$	X est un transversal de $\mathcal{H}$ et $\|X\|$ est minimale
	$\Leftrightarrow$	E_X est une couverture de $Q_{\mathcal{H}}$ en utilisant $\mathcal{T}_{\mathcal{H}}$ qui minimise la taille du rest (d'après les lemmes 3.1.1 et 3.3.1 (1)) et $\|X\|$ est minimale
	$\Leftrightarrow$	E_X est une couverture de $Q_{\mathcal{H}}$ en utilisant $\mathcal{T}_{\mathcal{H}}$ qui minimise la taille du rest et $\|Miss_{E_X}(Q_{\mathcal{H}})\|$ est minimale (d'après le lemme 3.3.1 (2))
	$\Leftrightarrow$	E_X est une meilleure couverture de $Q_{\mathcal{H}}$ en utilisant $\mathcal{T}_{\mathcal{H}}$

□

Ainsi on vient de montrer que le problème de calcul d'un transversal de cardinalité minimale pouvait se réduire dans celui des meilleures couvertures. Comme le premier est NP-Difficile [38, 32], alors le second l'est aussi. Ceci termine la preuve du théorème 3.3.1.

3.3.2 A propos de la complexité de l'algorithme *computeBCov*

Comme le problème de calcul des transversaux minimaux d'un hypergraphe est exponentiel au pire (en espace et donc en temps), il en est de même pour $\mathcal{BCOV}(\mathcal{T}, Q)$ et donc pour tout algorithme juste et complet qui l'implémente. Aller plus loin dans l'étude de complexité de *computeBCov* signifierait étudier la technique de Branch and Bound sur laquelle il repose. Or, celle-ci dépend fortement de l'heuristique d'évaluation du coût d'une solution en construction. Il est donc difficile d'avoir une idée précise de la complexité du Branch and Bound. Cela ne nous permet pas de donner plus de détails quant à la complexité de *computeBCov*.

Cependant, les expérimentations présentées dans l'annexe D, page 185, donnent un bon aperçu du comportement de *computeBCov*. Grâce à des tests sur des ontologies aléatoires, on montre l'effet du Branch and Bound limitant l'explosion combinatoire du nombre de transversaux générés au cours de l'algorithme. En ce qui concerne le calcul des transversaux minimaux, on rappelle qu'une étude détaillée est donnée dans l'annexe C, page 177, montrant comment les persistants permettent de diminuer le temps moyen de génération d'un transversal minimal, par rapport à l'algorithme classique. Ces conclusions sont résumées dans la section 7.2.2, page 122.

Chapitre 4

Une solution pour le langage $\mathcal{ALN}$

Sommaire

Au chapitre 3, nous avons vu, dans le cadre des langages ayant la propriété de subsomption structurelle, que trouver les meilleures couvertures d'un concept en utilisant une terminologie revient à résoudre un problème de transversaux minimaux de coût minimal dans un hypergraphe. La caractéristique de cette approche est que la propriété de subsomption structurelle limite l'expressivité des langages par une simplification des interactions entre constructeurs. Il en résulte une simplification dans la réécriture, comme notamment le fait que l'opérateur de différence sémantique soit calculable structurellement comme une différence ensembliste. Cependant, que devient le problème des meilleures couvertures quand le langage offre plus de constructeurs et donc d'interactions entre eux, comme c'est le cas dans les systèmes de raisonnements existants ou dans les langages clés du web sémantique ?

C'est la question à laquelle nous essayons de répondre de ce chapitre qui étudie le cas du langage $\mathcal{ALN}$, ni trop complexe pour les raisonnements standard, ni trop simple au niveau expressivité. Nous montrons que son expressivité implique de reformuler le problème pour prendre en compte les nouvelles interactions entre constructeurs, notamment celles qui aboutissent aux cas d'inconsistance. Dans l'esprit de "diviser pour régner", nous proposons un algorithme de résolution en plusieurs étapes, chacune correspondant à une réduction de l'espace de recherche.

Nous commençons par présenter plus en détails les difficultés nouvelles ainsi que l'approche de résolution proposée.

4.1 Aperçu de l'approche

Dans la découverte des meilleures couvertures, le langage $\mathcal{ALN}$ apporte les spécificités suivantes par rapport aux langages ayant la propriété de subsomption structurelle :

- L'expression de l'inconsistance de manière non triviale (par exemple l'inconsistance due à l'interaction des constructeur $\leq nR$ et $\geq mR$, pour $m > n$) : il faut éviter de générer des couvertures inconsistantes. De plus l'expression non triviale de l'inconsistance induit, comme on l'a vu dans la section 1.3, que la différence n'est pas unique. Il faut donc disposer d'un moyen pour calculer la différence dans ces cas-là. Il faut également redéfinir la notion de couverture pour tenir compte du fait que le résultat d'une différence n'est plus sémantiquement unique.
- Un critère d'optimalité sémantique : la différence n'étant pas unique, le rest d'une réécriture n'est pas unique non plus. Dès lors, comparer les réécritures d'après la taille de leur rest devient problématique. C'est pourquoi le critère d'optimalité qui était précédemment la minimisation des tailles des rest et miss devient la maximisation des rest et miss par rapport à la subsomption, car c'est le moyen sémantique de minimiser la quantité d'information contenue dans une description.

Ainsi, dans le contexte d'$\mathcal{ALN}$, le problème est plus complexe et nécessite d'être reformulé. Le nouvel algorithme qui en résulte est appelé *computeALNBCov*. A l'instar de *computeBCov*, l'espace de recherche reste le treillis des parties de l'ensemble $\mathcal{S}_\mathcal{T}$ des concepts définis S_i de $\mathcal{T}$, puisque l'on cherche toujours des conjonctions de S_i. Cependant, contrairement au chapitre 3 où tous les critères définissant les meilleures couvertures sont résolus par un seul et même calcul de transversaux minimaux de coût minimal, nous traitons ici les difficultés séparément. Ainsi, nous étudions l'une après l'autre chaque condition définissant les meilleures couvertures, ce qui permet de supprimer à chaque fois de l'espace de recherche les ensembles de S_i ne vérifiant pas la condition considérée. Concrètement, la démarche proposée est la suivante : nous caractérisons chaque condition et, à partir de cette caractérisation, nous déterminons quels sont les plus grands (respectivement petits) ensembles de S_i qui vérifient cette condition. On obtient ainsi des bornes supérieures (respectivement inférieures) des meilleures couvertures dans l'espace de recherche. Celui-ci est donc réduit progressivement, pour ne plus contenir, en fin de processus, que les ensembles de S_i qui correspondent aux meilleures couvertures.

Ce chapitre est organisé en quatre sections : formalisation, étude, résolution et complexité du problème de découverte des meilleures couvertures dans $\mathcal{ALN}$.

Section 4.2 : Formalisation du problème

- Section 4.2.1 : nous définissons une forme normale à base de clauses qui permet d'exprimer les propriétés importantes de l'étude en termes ensemblistes faciles à manipuler. Puis nous rappelons les caractérisations de la subsomption, du lcs et de tous les cas

d'inconsistance possibles dans $\mathcal{ALN}$, caractérisations exprimées dans la forme normale à base de clauses.
- Section 4.2.2 : nous caractérisons la différence sémantique de Teege pour $\mathcal{ALN}$ et donnons un moyen de la calculer dans ce contexte. A la suite, nous donnons une définition de la notion de subsomption entre ensembles de descriptions utile pour comparer les résultats des différences sémantiques.
- Section 4.2.3 : nous définissons les notions de couvertures et de meilleures couvertures d'un $\mathcal{ALN}$-concept étant donnée une $\mathcal{ALN}$-terminologie, ainsi que le problème $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$ de découverte des meilleures couvertures dans $\mathcal{ALN}$.

Section 4.3 : Etude du problème $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$
- Sections 4.3.1, 4.3.2, 4.3.3 et 4.3.4 : nous étudions et caractérisons en termes de clauses chaque condition définissant les meilleures couvertures. Nous montrons comment ces caractérisations impliquent la découverte de bornes supérieures et inférieures réduisant l'espace de l'espace de recherche.

Section 4.4 : calcul des meilleures couvertures
- Sections 4.4.1 et 4.4.2 : nous présentons le calcul de tous les cas possibles d'inconsistance. Ceci débouche sur l'obtention d'une borne supérieure et d'une borne inférieure dans l'espace de recherche des meilleures couvertures.
- Sections 4.4.3 et 4.4.4 : nous présentons le calcul des couvertures de rest maximal par rapport à la subsomption. Ceci induit l'amélioration des bornes supérieure et inférieure. Nous montrons alors comment obtenir les meilleures couvertures.
- Section 4.4.5 : l'algorithme *compute*$\mathcal{ALN}$*BCov* est donné.

Section 4.5 : complexité des meilleures couvertures
- Section 4.5.1 : nous montrons que $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$ est un problème NP-Difficile.
- Section 4.5.2 : nous donnons et discutons la complexité de chaque étape de l'algorithme *compute*$\mathcal{ALN}$*BCov*.

L'étude des meilleures couvertures pour $\mathcal{ALN}$ est illustrée par un exemple détaillé appelé exemple "courant". De plus, ce chapitre étant assez long et technique, les démonstrations sont reportées en annexe.

4.2 Formalisation des meilleures couvertures dans $\mathcal{ALN}$

Dans cette section, nous définissons le problème de découverte des meilleures couvertures dans $\mathcal{ALN}$. Nous commençons dans la section 4.2.1 par quelques préliminaires indispensables à l'étude des meilleures couvertures. Dans la section 4.2.2, nous donnons une caractérisation de la différence sémantique dans $\mathcal{ALN}$. Et nous terminons par la définition des notions de couverture et de meilleure couverture dans la section 4.2.3.

4.2.1 Préliminaires

Forme normale à base de clauses

A l'instar du chapitre précédent concernant les langages ayant la propriété de subsomption structurelle, nous allons mener l'étude des meilleures couvertures en utilisant une forme normale basée sur la notion de clause, redéfinie dans le cas du langage $\mathcal{ALN}$. Cela revient à travailler sur les termes de la conjonction de la forme normalisée des descriptions.

Les règles de normalisation que l'on utilise sont bien connues dans les logiques de description (voir [6, 9, 51]) : leur rôle est principalement d'expliciter tous les cas d'inconsistances (i.e. où une

conjonction de deux termes est équivalente au concept inconsistant $\bot$) et d'oter les redondances des descriptions. Ces règles sont les suivantes (avec E et F des noms de concepts et A un concept atomique) :

- $E \sqcap \top \xrightarrow{\equiv} E$,
- $E \sqcap E \xrightarrow{\equiv} E$,
- $A \sqcap \neg A \xrightarrow{\equiv} \bot$,
- $E \sqcap \bot \xrightarrow{\equiv} \bot$,
- $(\forall R.\top) \xrightarrow{\equiv} \top$,
- $(\forall R.E) \sqcap (\forall R.F) \xrightarrow{\equiv} (\forall R.(E \sqcap F))$,
- $(\forall R.\bot) \xrightarrow{\equiv} (\leq 0\ R)$,
- $(\geq 0\ R) \xrightarrow{\equiv} \top$,
- $(\exists R) \xrightarrow{\equiv} (\geq 1\ R)$,
- $(\geq n\ R) \sqcap (\leq m\ R) \xrightarrow{\equiv} \bot$ si $m < n$ et
- $(\geq n\ R) \sqcap (\geq m\ R) \xrightarrow{\equiv} (\geq n\ R)$ si $m < n$.
- $(\leq n\ R) \sqcap (\leq m\ R) \xrightarrow{\equiv} (\leq m\ R)$ si $m < n$.

On dit qu'une $\mathcal{ALN}$-description C est normalisée si les règles précédentes lui ont été appliquées exhaustivement. On note $\widehat{C}$ la description normalisée de C.

La démarche qui consiste à travailler sur les termes de la conjonction constituant la description normalisée est connue dans la littérature [39]. Nous l'utilisons en lui donnant ici le nom de forme normale à base de clauses, où une clause est un terme d'une conjonction (pas forcément normalisée). Notons que cette notion de clause diffère de celle du chapitre 3.

Définition 4.2.1 ($\mathcal{ALN}$-clause et forme normale à base de clauses)
Une $\mathcal{ALN}$-clause, ou plus simplement clause, d'une $\mathcal{ALN}$-description C est un terme de la conjonction obtenue à partir de C après application exhaustive de la règle suivante $(\forall R.(E \sqcap F)) \xrightarrow{\equiv} (\forall R.E) \sqcap (\forall R.F)$. *L'ensemble des clauses de C est noté $C^{\#}$. Une clause de la forme $\forall R_1.(...(\forall R_n.P)...)$, avec P un nom de concept atomique, une négation de concept atomique ou une restriction numérique, sera notée de manière plus concise $\forall R_1...R_n.P$.*

L'ensemble des clauses de la description normalisée constitue ce qu'on appelle la forme normale à base de clauses. Elle est donc obtenue en appliquant exhaustivement les règles de normalisation, puis exhaustivement la règle $(\forall R.(E \sqcap F)) \xrightarrow{\equiv} (\forall R.E) \sqcap (\forall R.F)$. *On note $\widehat{\mathcal{G}}_C^{\#}$ cette forme normale à base de clauses de la description C*[13].

Pour bien différencier les notations précédentes, voyons un exemple.

Exemple 13
Soit l'$\mathcal{ALN}$-description $C \equiv \forall T.((\geq 4\ R) \sqcap (\exists S) \sqcap (\leq 1\ R)) \sqcap (\geq 2\ R) \sqcap (\forall Q.A) \sqcap (\forall Q.(\forall R.(\forall S.(B \sqcap \neg B))))$. On a alors :

$C^{\#} = \{\forall T.\geq 4\ R, \forall T.\exists S, \forall T.\leq 1\ R, \geq 2\ R, \forall Q.A, \forall QRS.B, \forall QRS.\neg B\}$,
$\widehat{C} = (\leq 0\ T) \sqcap (\geq 2\ R) \sqcap (\forall Q.(A \sqcap \forall R.(\leq 0\ S)))$,
$\widehat{\mathcal{G}}_C^{\#} = \{\leq 0\ T, \geq 2\ R, \forall Q.A, \forall QR.\leq 0\ S\}$.

On voit ici que la différence entre $C^{\#}$ et $\widehat{\mathcal{G}}_C^{\#}$ est la nature normalisée des clauses de $\widehat{\mathcal{G}}_C^{\#}$. En effet, $\widehat{\mathcal{G}}_C^{\#}$ est obtenue à partir d'une description normalisée $\widehat{C}$ dans laquelle toutes les inconsistances ont été explicitées et les clauses les moins spécifiques supprimées. Par exemple, la clause $\forall T.(\exists S)$

[13] On utilise la notation $\widehat{\mathcal{G}}_C^{\#}$ et non $\widehat{C^{\#}}$ qui pourrait sembler plus logique, car pour des descriptions C complexes et longues à écrire, le résultat est difficile à lire et à comprendre. Par exemple, $\widehat{\mathcal{G}}^{\#}_{A \sqcap B \sqcap ... \sqcap Z}$ est plus clair que $\widehat{A \sqcap B \sqcap ... \sqcap Z}^{\#}$.

de C est supprimée dans $\widehat{\mathcal{G}}_C^{\#}$ par la clause $\leq 0\ T$ qui est elle-même le résultat d'une inconsistance entre les clauses $\forall T.(\leq 1\ R)$ et $\forall T.(\geq 4\ R)$ de C. ◦

Notons que, pour une $\mathcal{ALN}$-description C, la construction de $\widehat{C}$ a été montrée polynomiale en la taille de C dans [19]. Donc on peut obtenir $\widehat{\mathcal{G}}_C^{\#}$ en temps polynomial en la taille de C.

L'intérêt d'utiliser cette forme normale à base de clauses est de travailler à un niveau de granularité très fin. Ceci permet d'exprimer certains résultats en termes ensemblistes faciles à appréhender et à manipuler. De plus, cela permet de réutiliser conjointement des résultats déjà démontrés avec deux formalismes très différents que sont les graphes de description [63, 50] et la théorie des automates [4, 50]. En effet, il est souvent simple de traduire en termes de clauses les résultats issus de ces deux formalismes. C'est précisément ce que l'on fait aux sections 4.2.1 et 4.2.2 avec le rappel des caractérisations pour $\mathcal{ALN}$ de la subsomption structurelle, du lcs et des inconsistances, qui sont des préalables à l'étude des meilleures couvertures. De plus, la notion de clause est très proche de l'implémentation et donc la facilite.

Subsomption structurelle et lcs avec les clauses

Dans cette section, nous donnons les caractérisations en termes de clauses de la subsomption et du plus petit subsumant commun (lcs). Ces résultats ne sont pas nouveaux puisqu'ils sont simplement la traduction avec les clauses des caractérisations déjà existantes de la subsomption (voir [63, 50]) et du plus petit subsumant commun (voir [50]).

On rappelle que l'on suppose que ces deux raisonnements se font par rapport à une terminologie acyclique $\mathcal{T}$ de définitions de concepts $\mathcal{ALN}$ dépliées et normalisées. C'est pourquoi on omet l'indice $_{\mathcal{T}}$ dans les notations $\sqsubseteq$, $\equiv$ et lcs.

Le théorème suivant donne la caractérisation à base de clauses de la subsomption structurelle dans $\mathcal{ALN}$.

Théorème 4.2.1 (Subsomption structurelle)
Soient C et D deux descriptions de concepts $\mathcal{ALN}$. On a :

$$C \sqsubseteq D \Leftrightarrow \forall c_D \in \widehat{\mathcal{G}}_D^{\#}, \exists c_C \in \widehat{\mathcal{G}}_C^{\#} \mid c_C \sqsubseteq c_D$$

Ce résultat est la traduction en termes de clauses des caractérisations structurelles de la subsomption données dans [63, 5] et exprimées avec les graphes de descriptions[14]. Un graphe de description permet de représenter une description sous forme graphique avec des nœuds et des arcs. Après normalisation des graphes de description par l'application de règles équivalentes à celles de la section 4.2.1 (voir [63]), la subsomption est caractérisée avec la notion de chemin fondamental qui représente une partie d'une description (voir [63, 5] pour plus de détails sur les chemins fondamentaux). On peut facilement passer de la notion de chemin fondamental à celle de clause, et la propriété caractérisant la subsomption avec les chemins fondamentaux se traduit alors par le fait que chaque clause de la description subsumante subsume une clause de la description subsumée.

On peut remarquer que cette caractérisation nécessite d'avoir normalisé les deux descriptions, alors que ce n'est pas nécessaire dans le théorème original de [63, 5] (seule C doit être normalisée). Sachant que les résultats ultérieurs (et les démonstrations) nécessaires à la découverte

[14] Rappelons que la notion de clause ici utilisée est différente de celle du chapitre 3. Ceci implique que le théorème 4.2.1 n'est pas équivalent à la propriété de subsomption structurelle. Autrement dit, $\mathcal{ALN}$ n'est pas un langage ayant la propriété de subsomption structurelle au sens de Teege.

des meilleures couvertures sont exprimés avec des concepts normalisés, et sachant que certains d'entre eux (caractérisation du lcs par exemple) ne s'appliquent que sur des concepts normalisés, l'hypothèse supplémentaire dans la caractérisation de la subsomption n'est pas gênante.

Basée sur la complexité de la normalisation, qui est polynomiale en fonction de la taille de la description, la complexité de la subsomption structurelle exprimée ainsi est aussi polynomiale en fonction des tailles des descriptions concernées.

Le lemme suivant dérive du théorème 4.2.1 et caractérise le plus petit subsumant commun.

Lemme 4.2.1 (Plus petit subsumant commun dans $\mathcal{ALN}$)
Soient A et B deux descriptions de concepts $\mathcal{ALN}$. On a :

$$\widehat{\mathcal{G}}^{\#}_{lcs(A,B)} = \{c_1 \mid (c_1, c_2) \in (\widehat{\mathcal{G}}^{\#}_A \times \widehat{\mathcal{G}}^{\#}_B) \cup (\widehat{\mathcal{G}}^{\#}_B \times \widehat{\mathcal{G}}^{\#}_A) \text{ et } c_2 \sqsubseteq c_1\}$$

Ce lemme est la traduction en termes de clauses du théorème 5.2.1 de [50]. L'existence et la complexité du plus petit subsumant commun est aussi donnée dans [50] : le lcs de deux $\mathcal{ALN}$-descriptions existe toujours et peut être calculé en temps polynomial en les tailles des deux descriptions.

Caractérisation des inconsistances

Dans cette section on s'intéresse à la caractérisation des inconsistances dans une $\mathcal{ALN}$-description.

Il est montré dans [5] que la normalisation d'une $\mathcal{ALN}$-description C fait apparaître dans la description normalisée tous les cas d'inconsistance. Trouver tous ces cas dans une $\mathcal{ALN}$-description est donc un problème résolu par la normalisation. Par contre, à partir d'un ensemble de concepts définis S_i, trouver a priori tous les cas d'inconsistances dans toute description construite comme conjonction de S_i, et ce sans être obligé de former toutes les conjonctions possibles de S_i et de les normaliser, est un problème encore non résolu à notre connaissance. Or, il se trouve qu'il se pose dans la découverte des meilleures couvertures (et dans le calcul de la différence sémantique). Pour aider à résoudre ce problème, nous rappelons les cas d'inconsistance existants dans $\mathcal{ALN}$ à la définition 4.2.2 et leur caractérisation au lemme 4.2.2.

Au sein d'une description, deux types d'inconsistance peuvent survenir. Les inconsistances explicites sont des ensembles de clauses dont la conjonction est inconsistante, rendant ainsi la description entière inconsistante. Les inconsistances implicites sont des inconsistances explicites apparaissant à une certaine profondeur dans l'enchaînement des rôles du constructeur $\forall$.

Définition 4.2.2 (Inconsistances explicite et implicite, et clause d'exclusion)
Soit une $\mathcal{ALN}$-description C.

- *C possède une inconsistance explicite si et seulement si $C \equiv \bot$ et $\forall c_i \in C^{\#},\ c_i \neq \bot$.*
- *C possède une inconsistance implicite aboutissant à la clause $c_\bot \in \widehat{\mathcal{G}}^{\#}_C$ de la forme $c_\bot = \forall R_1...R_{i-1}.(\leq 0\ R_i)$ (avec $i \geq 1$) si et seulement si :*

$$\exists S \subseteq C^{\#} \mid c_\bot = \widehat{\sqcap_{c \in S} c} \text{ et } |S| \geq 2$$

(i.e. $c_\bot$ est obtenue par normalisation de la conjonction d'un ensemble $E_\bot$ d'au moins 2 clauses de C).
Une telle clause $c_\bot$ est nommée clause d'exclusion[15] *de C.*

[15] Le mot $R_1...R_i$ de la clause $c_\bot = \forall R_1...R_i.\bot$ résultat d'une inconsistance implicite est appelé un mot d'exclusion de C [49, 5, 4, 50] (ou "*C-excluding word*" en anglais). En nommant ${Inc_Q}^C$ l'ensemble des clauses d'exclusion de C, on notera qu'il y a une différence entre ${Inc_Q}^C$, qui est l'ensemble des clauses issues d'une inconsistance implicite dans C, et l'ensemble E_C des mots d'exclusion de C défini dans [50] qui regroupe tous les mots $w = R_1R_2...R_n$ tels que $C \sqsubseteq \forall w.\bot$. On a cependant ${Inc_Q}^C \subseteq \{\forall w.\bot, w \in E_C\}$.

Exemple 14
La description $D \equiv \forall R_1R_2.P \sqcap \forall R_1R_2.\neg P \sqcap \forall R_1. \geq 3R_2 \sqcap \geq 4R_1$ possède une inconsistance explicite puisque $D \equiv \bot$. En effet, $\forall R_1R_2.P \sqcap \forall R_1R_2.\neg P \equiv \forall R_1. \leq 0R_2$, puis $\forall R_1. \leq 0R_2 \sqcap \forall R_1. \geq 3R_2 \equiv \leq 0R_1$, et enfin $\leq 0R_1 \sqcap \geq 4R_1 \equiv \bot$.
La description $D' \equiv \forall R_1R_2.P \sqcap \forall R_1R_2.\neg P \sqcap \forall R_1. \geq 3R_2$ possède une inconsistance implicite puisque $D' \equiv \forall R_1.\bot$. En effet, $\forall R_1R_2.P \sqcap \forall R_1R_2.\neg P \equiv \forall R_1. \leq 0R_2$, puis $\forall R_1. \leq 0R_2 \sqcap \forall R_1. \geq 3R_2 \equiv \leq 0R_1 \equiv \forall R_1.\bot$. ∘

Ainsi, une inconsistance (explicite ou implicite) peut être engendrée par la conjonction d'au moins deux clauses. Il est donc nécessaire de caractériser les ensembles de clauses engendrant des inconsistances dans les descriptions. Basé sur les lemmes 4.2.2 et 6.1.4 de [50], le lemme suivant utilise l'approximation faible pour caractériser les cas d'inconsistance. L'approximation faible permet d'exprimer cette caractérisation de manière concise (par rapport aux lemmes de [50]), et apporte en plus une propriété de maximalité par rapport à la subsomption indispensable au calcul de la différence sémantique dans $\mathcal{ALN}$(voir le théorème 4.2.2 et sa démonstration dans l'annexe A.4 page 150).

Lemme 4.2.2 (Caractérisation des inconsistances)
Soit C une $\mathcal{ALN}$-description de concept.

(i) $C \equiv \bot \quad \Leftrightarrow \quad \exists c \in C^{\#} \mid \forall c_{\uparrow} \in \widehat{\mathcal{G}}^{\#}_{Approx_{\uparrow}(\neg c)},\ \exists c' \in C^{\#} \mid\ (c' \sqsubseteq c_{\uparrow}\ et\ prof(c') = prof(c_{\uparrow}))$

(ii) *C présente une inconsistance implicite* $\Leftrightarrow$ $\exists c \in C^{\#} \mid$
- $c = \forall R_1R_2...R_{i-1}.c^*$, *avec* $i \geq 2$,
- $\forall c_{\uparrow} \in \widehat{\mathcal{G}}^{\#}_{Approx_{\uparrow}(\neg c^*)}, \exists c' \mid \forall R_1R_2...R_{i-1}.c' \in C^{\#}$ *et* $c' \sqsubseteq c_{\uparrow}$ *et* $prof(c') = prof(c_{\uparrow})$
- $\forall R_1R_2...R_{i-2}.(\geq p_{i-1}.R_{i-1}) \notin C^{\#}, \forall p_{i-1} \geq 1$

Dans le lemme précédent, le point (i) caractérise les inconsistances explicites dans $\mathcal{ALN}$, et le point (ii) les inconsistances implicites. Dans le cas d'inconsistance implicite, la clause d'exclusion $c_{\bot}$ obtenue est :
$c_{\bot} \equiv c \sqcap (\sqcap_{c'} \forall R_1R_2...R_{i-1}.c') \equiv \forall R_1R_2...R_{i-2}.(\leq 0\ R_{i-1})$

La démonstration de ce lemme est donnée dans l'annexe A.3 page 145. Son principe est de constater qu'une description C qui contient une inconsistance explicite contient toujours au moins une clause c inconsistante par conjonction avec un ensemble d'autres clauses de la description. Or pour obtenir $\bot$ comme conjonction de c avec un ensemble de clauses, il faut a priori que cet ensemble soit équivalent à $\neg c$. Comme $\mathcal{ALN}$ ne possède pas la négation complète, il est naturel d'utiliser l'approximation de la négation de c. Or $\neg c$ est une $\mathcal{ALEN}$-description pour laquelle la notion d'approximation faible est définie dans [39, 41] et rappelée dans la section 1.2.2 page 13. On est donc en mesure de calculer cette approximation. On démontre alors que toute clause de l'ensemble des clauses devant être ajoutées à c pour obtenir $\bot$ doit être subsumée par une clause de cette approximation. Pour une inconsistance implicite, c'est le même principe, sauf que l'inconsistance a lieu a une certaine profondeur dans l'enchaînement des rôles du constructeur $\forall$, et qu'il faut assurer qu'on n'a pas une inconsistance explicite.

Exemple 15
Dans l'exemple précédent, on a $D \equiv \bot$ car la clause $c_D = \forall R_1R_2.P$ est inconsistante par conjonction avec l'ensemble des autres clauses de D qui sont $\forall R_1R_2.\neg P$, $\forall R_1. \geq 3R_2$ et $\geq 4R_1$. Si on calcul l'approximation faible de $\neg c_D \equiv \exists R_1.\exists R_2.\neg P$, on obtient les clauses suivantes

$\forall R_1R_2.\neg P$, $\forall R_1. \geq 1R_2$ et $\geq 1R_1$. On constate bien que chaque clause restante de D est subsumée par une clause de l'approximation.
Pour D', comme $D' \equiv \forall R_1.\bot$, on se place à une profondeur de 1 (après R_1), ce qui signifie qu'on va cherche une inconsistance explicite parmi les clauses de D' après avoir enlevé le premier rôle de chaque clause. On constate la clause $\forall R_2.P$ est en inconsistance avec les clauses $\forall R_2.\neg P$ et $\geq 3R_2$. Ceci s'obtient en calculant l'approximation faible de la négation de $\forall R_2.P$, composée des clauses $\forall R_2.\neg P$ et $\geq 1R_2$, et en constatant que ces deux clauses subsument chacune une clause de D' restante en étant de même profondeur. Comme il n'y a pas d'autre clause induisant une inconsistance explicite dans D', on a bien une inconsistance implicite. ◦

4.2.2 La différence sémantique dans $\mathcal{ALN}$

Dans cette section, nous rappelons la définition de la différence sémantique de Teege [74], puis nous donnons sa caractérisation dans $\mathcal{ALN}$.

Définition 4.2.3 (Différence sémantique pour $\mathcal{ALN}$ [74])
Soient deux $\mathcal{ALN}$-descriptions de concepts A et B. Pour $B \sqsubseteq A$, on définit :
$B - A = Max_{\sqsubseteq}\{C_i | C_i \sqcap A \equiv B\}$ *où les C_i sont des $\mathcal{ALN}$-descriptions.*

La difficulté du calcul d'une différence $B - A$ vient des cas de "recompositions" du concept $\bot$ dans une clause de B à partir d'une clause de A et d'une ou plusieurs clauses des C_i à construire.

Premièrement, ceci implique que $B - A$ peut être un ensemble de descriptions non sémantiquement équivalentes, comme cela est montré dans l'exemple suivant.

$$\forall R_1.\bot - \forall R_1.(P \sqcap (\geq 3\ R_2) \sqcap (\leq 0\ R_3)) = \{C_1 = \forall R_1.\neg P,\ C_2 = \forall R_1.(\leq 2\ R_2),\ C_3 = \forall R_1.(\geq 1\ R_3)\}$$

Il est facile de voir que l'exemple précédent donne bien les trois seules descriptions C_i qui sont maximales par rapport à la subsomption telles que $C_i \sqcap A \equiv B$, pour $i = 1$, $i = 2$ et $i = 3$.

Deuxièmement, si $\widehat{\mathcal{G}}^{\#}_B$ possède la clause $\forall R_1...R_{n-1}. \leq 0\ R_n$ (i.e. $\forall R_1...R_n.\bot$) et $\widehat{\mathcal{G}}^{\#}_A$ la clause $\forall R_1...R_nR_{n+1}...R_m.P$, alors, d'après le lemme 4.2.2, les $\widehat{\mathcal{G}}^{\#}_{C_i}$ devraient posséder des clauses subsumées par celles de l'approximation faible de $\forall R_1...R_n.(\neg(\forall R_{n+1}...R_m.P))$, et de même profondeur. Comme on veut des descriptions maximales par rapport à la subsomption, il faut que les $\widehat{\mathcal{G}}^{\#}_{C_i}$ possèdent directement les clauses de cette l'approximation faible, puisque l'approximation faible assure la maximalité par rapport à la subsomption. En suivant ce principe, le théorème 4.2.2 montre comment calculer une description résultat d'une différence.

Théorème 4.2.2 (Calcul d'une différence sémantique dans $\mathcal{ALN}$)
Soient deux $\mathcal{ALN}$-descriptions A et B telles que $B \sqsubseteq A$. Une $\mathcal{ALN}$-description $C \in B - A$ est construite ainsi : les seules clauses de $\widehat{\mathcal{G}}^{\#}_C$ sont construites de la manière suivante :

$\forall c_B \in \widehat{\mathcal{G}}^{\#}_B$

- *Si (cas des inconsistances) :*
  $c_B = \forall R_1R_2...R_{n-1}.(\leq 0\ R_n)$, *avec* $n \geq 0$ *(si* $n = 0$ *alors* $c_B = \bot$*)*
  $\exists c_A \in \widehat{\mathcal{G}}^{\#}_A \mid c_A = \forall R_1R_2...R_nR_{n+1}...R_{n+m}.P$, $m \geq 0$
  alors, on ajoute à $\widehat{\mathcal{G}}^{\#}_C$ *toutes les clauses c vérifiant :*
  $\left(c = \forall R_1R_2...R_n.c', c' \in \widehat{\mathcal{G}}^{\#}_{Approx_{\uparrow}(\neg\forall R_{n+1}...R_{n+m}.P)}\right)$ *et* $(A \not\sqsubseteq c)$
- *Sinon (cas général) : si* $c_B \notin \widehat{\mathcal{G}}^{\#}_A$, *alors on ajoute* c_B *à* $\widehat{\mathcal{G}}^{\#}_C$.

$\widehat{\mathcal{G}}_B^{\#}$	$\widehat{\mathcal{G}}_A^{\#}$	$\widehat{\mathcal{G}}_{C_1}^{\#}$	$\widehat{\mathcal{G}}_{C_2}^{\#}$	$\widehat{\mathcal{G}}_{C_3}^{\#}$
P_1	P_1			
$\leq 1\ S$	$\leq 2\ S$	$\leq 1\ S$	$\leq 1\ S$	$\leq 1\ S$
$\forall T.P_1$	$\forall T.P_1$			
$\forall TV.P_3$		$\forall TV.P_3$	$\forall TV.P_3$	$\forall TV.P_3$
$\forall S.(\geq 6\ R)$	$\forall S.(\geq 3\ R)$	$\forall S.(\geq 6\ R)$	$\forall S.(\geq 6\ R)$	$\forall S.(\geq 6\ R)$
$\forall S.(\leq 0\ T)$	$\forall S.(\leq 6\ T)$			
	$\forall ST.P_2$	$\forall ST.(\neg P_2)$		
	$\forall STV.(\geq 4\ W)$		$\forall STV.(\leq 3\ W)$	
			$\forall ST.(\geq 1\ V)$	
	$\forall STVWX.(\leq 3\ R)$			$\forall STVWX.(\geq 4\ R)$
				$\forall STVW.(\geq 1\ X)$
				$\forall ST.(\geq 1\ V)$

Tab. 4.1 – Un exemple de calcul d'une différence sémantique.

La démonstration de ce théorème est donnée dans l'annexe A.4 page 150. La construction d'une solution $C \in B - A$ est basée sur le fait d'obtenir dans $\widehat{\mathcal{G}}_{C \sqcap A}^{\#}$ exactement toutes les clauses de $\widehat{\mathcal{G}}_B^{\#}$. Ainsi le premier cas est celui où la clause de B peut être obtenue par inconsistance, et le second cas (cas général) est celui où l'on est obligé de garder dans C la clause de B. L'argument principal permettant de prouver la justesse et la complétude de ce théorème est le fait que l'approximation faible dans $\mathcal{ALN}$ d'une $\mathcal{ALEN}$-description est maximale par rapport à la subsomption. Notons que les multiples descriptions résultats d'une différence proviennent du cas des inconsistances pouvant survenir pour plusieurs clauses c_B de B.

Exemple 16

Soient les $\mathcal{ALN}$-descriptions B et A dont les ensembles de descriptions normalisées sont donnés dans le tableau 4.1. On peut remarquer que $B \sqsubseteq A$. Le résultat de la différence $B - A$ est l'ensemble des 3 descriptions C_1, C_2 et C_3 sémantiquement non équivalentes et incomparables. Le tableau 4.1 détaille les trois solutions. Ces 3 solutions viennent de la clause $\forall S.(\leq 0\ T)$ de $\widehat{\mathcal{G}}_B^{\#}$ induisant trois cas d'inconsistance avec les clauses $\forall ST.P_2$, $\forall STV.(\geq 4\ W)$ et $\forall STVWX.(\leq 3\ R)$ de $\widehat{\mathcal{G}}_A^{\#}$. Par exemple, pour obtenir $\forall S.(\leq 0\ T)$ à partir $\forall STVWX.(\leq 3\ R)$, il faut ajouter (par conjonction) à $\forall STVWX.(\leq 3\ R)$ l'ensemble de clauses :

$$\{c = \forall S.c' \mid c' \in \widehat{\mathcal{G}}^{\#}_{Approx_{\uparrow}(\neg \forall TVWX.(\leq 3\ R))}\} = \{\forall STVWX.(\geq 4\ R), \forall STVW.(\geq 1\ X), \forall STV.(\geq 1\ W), \forall ST.(\geq 1\ V)\}$$

Comme $\widehat{\mathcal{G}}_A^{\#}$ possède déjà $\forall STV.(\geq 4\ W)$ (et donc $A \sqsubseteq \forall STV.(\geq 1\ W)$), alors on limite l'ensemble précédent à $\{\forall STVWX.(\geq 4\ R), \forall STVW.(\geq 1\ X), \forall ST.(\geq 1\ V)\}$.

La clause $\forall S.(\leq 6\ T)$ de $\widehat{\mathcal{G}}_A^{\#}$ n'induit pas un cas d'inconsistance avec $\forall S.(\leq 0\ T)$ de $\widehat{\mathcal{G}}_B^{\#}$, car leur profondeur est la même. Les autres clauses de $\widehat{\mathcal{G}}_B^{\#}$ sont traitées par le cas général : par exemple, la clause $\leq 1\ S$ n'est pas dans $\widehat{\mathcal{G}}_A^{\#}$, donc elle fait partie de toutes les solutions, ce qui n'est pas le cas de la clause $\forall T.P_1$ qui est dans $\widehat{\mathcal{G}}_A^{\#}$.

∘

Nous introduisons maintenant la notion de subsomption d'un ensemble d'$\mathcal{ALN}$-descriptions par un autre ensemble d'$\mathcal{ALN}$-descriptions. Cette extension de l'opérateur de subsomption à des ensembles de descriptions est motivée par le fait que l'on veut pouvoir comparer les résultats

issus de la différence sémantique. Comme le résultat d'une différence peut être un ensemble de descriptions, alors il nous faut un opérateur de comparaison entre ensembles de descriptions. Comme on veut pouvoir continuer à utiliser un critère sémantique (comme dans la différence sémantique), alors il semble naturel d'étendre la notion de subsomption à des ensembles de clauses.

Définition 4.2.4 (Subsomption entre ensembles de descriptions)
Soient $\mathcal{C}_1 = \{C_i^1, i \in \{1,..,n\}\}$ et $\mathcal{C}_2 = \{C_j^2, j \in \{1,..,m\}\}$ deux ensembles de descriptions de concepts $\mathcal{ALN}$ (dans chaque ensemble, les concepts sont incomparables deux à deux par rapport à la subsomption). On dit que $\mathcal{C}_1$ est subsumé par $\mathcal{C}_2$, et on note $\mathcal{C}_1 \sqsubseteq \mathcal{C}_2$, si et seulement si :

$$\forall C_i^1 \in \mathcal{C}_1, \exists C_j^2 \in \mathcal{C}_2 \mid C_i^1 \sqsubseteq C_j^2$$

L'equivalence est alors définie de la manière suivante :

$$\mathcal{C}_1 \equiv \mathcal{C}_2 \Leftrightarrow \mathcal{C}_1 \sqsubseteq \mathcal{C}_2 \text{ et } \mathcal{C}_2 \sqsubseteq \mathcal{C}_1$$

On peut remarquer que cette définition correspond à la subsomption classique pour deux ensembles de descriptions composés d'une seule description, puisque pour deux $\mathcal{ALN}$-descriptions C et D on a $\{C\} \sqsubseteq \{D\} \Leftrightarrow C \sqsubseteq D$. Le symbole de subsomption reste le même car le contexte de son utilisation lèvera toujours l'ambiguïté.

4.2.3 Définition du problème $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$

Dans cette section nous donnons la définition des couvertures d'une $\mathcal{ALN}$-description de concept étant donnée une terminologie $\mathcal{T}$, puis celle des meilleures couvertures. Ensuite, au travers d'exemples, nous illustrons ces définitions.

Définition 4.2.5 (Couvertures d'un concept $\mathcal{ALN}$ en utilisant une terminologie)
Soit $\mathcal{T}$ une $\mathcal{ALN}$-terminologie composée uniquement d'axiomes terminologiques de définitions de concept, et $Q \not\equiv \bot$ une $\mathcal{ALN}$-description de concept consistante. L'ensemble de tous les noms de concepts définis consistants S_i de $\mathcal{T}$ est noté $\mathcal{S}_\mathcal{T} = \{S_i, i \in \{1,..,n\}\}$ avec $S_i \not\equiv \bot, \forall i \in \{1,n\}$. Une couverture $\mathcal{ALN}$ E de Q en utilisant $\mathcal{T}$ est une $\mathcal{ALN}$-description de concept telle que :

a) E est une conjonction de noms de concepts définis S_i de $\mathcal{S}_\mathcal{T}$:
$E \equiv S_{i_1} \sqcap S_{i_2} \sqcap ... \sqcap S_{i_p}$ avec $i_j \in \{1,..,n\}$ et $\forall j, j \in \{1,..,p\}$,

b) $Q \sqcap E \not\equiv \bot$ et

c) $Q - lcs(Q, E) \not\equiv \{Q\}$.

On pourra dans la suite identifier E soit comme la conjonction de ses S_i, soit comme l'ensemble de ses S_i (le contexte lèvera les éventuelles ambiguités).

On nomme $cov(\mathcal{T}, Q)$ l'ensemble des couvertures de Q en utilisant $\mathcal{T}$.

La condition (a) définit le langage dans lequel les couvertures de Q sont exprimées : celui-ci est $\{\sqcup, \sqcap\}$ car il y a éventuellement plusieurs couvertures à Q, et chacune d'entre elles est une conjonction de S_i. La condition (b) assure la consistance de la couverture par rapport au concept Q à couvrir. La condition (c) définit la notion de couverture : on considère que E est une couverture de Q si E et Q partagent des informations, autrement dit si certaines informations se trouvant dans Q se trouvent aussi dans E.

Nous définissons maintenant les meilleures couvertures d'une $\mathcal{ALN}$-description de concept étant donnée une terminologie $\mathcal{T}$.

Définition 4.2.6 (Meilleures couvertures $\mathcal{ALN}$ en utilisant une terminologie)
Soit $\mathcal{T}$ une $\mathcal{ALN}$-terminologie composée uniquement d'axiomes terminologiques de définitions de concept, et $Q \not\equiv \bot$ une $\mathcal{ALN}$-description de concept. L'ensemble de tous les noms de concepts définis consistants S_i de $\mathcal{T}$ est noté $\mathcal{S}_\mathcal{T}$= $\{S_i, i \in \{1,..,n\}\}$ avec $S_i \not\equiv \bot, \forall i \in \{1,n\}$. Une meilleure couverture $\mathcal{ALN}$ E de Q en utilisant $\mathcal{T}$ est une $\mathcal{ALN}$-description de concept telle que :

a,b,c) E est une couverture $\mathcal{ALN}$ de Q utilisant $\mathcal{T}$,

d) $Rest_E(Q)$ doit être maximum par rapport à la subsomption (entre ensembles de descriptions), avec
$Rest_E(Q) := Q - lcs(Q, E)$,

e) à rest équivalents, $Miss_E(Q)$ doit être maximum par rapport à la subsomption (entre ensembles de descriptions), avec
$Miss_E(Q) := E - lcs(Q, E)$

f) à rest et miss équivalents, E (considérée comme ensemble de S_i) doit être minimale par rapport à l'inclusion.

On nomme $cov_{rest}(\mathcal{T}, Q)$ l'ensemble des couvertures de Q en utilisant $\mathcal{T}$ maximisant le rest par rapport à la subsomption.

On nomme $bcov(\mathcal{T}, Q)$ l'ensemble des meilleures couvertures de Q en utilisant $\mathcal{T}$.

Le problème de découverte des meilleures couvertures dans $\mathcal{ALN}$ est celui du calcul de toutes les meilleures couvertures $\mathcal{ALN}$ E de Q en utilisant $\mathcal{T}$, et est noté $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$ (pour "$\mathcal{ALN}$-best covering problem" en anglais).

Les conditions (d) et (e) définissent les critères à maximiser pour être une meilleure couverture de Q : $Rest_E(Q)$ donne l'information présente dans Q et absente de la couverture E et $Miss_E(Q)$ donne l'information présente dans E et absente de Q. Maximiser ces deux ensembles de descriptions par rapport à la subsomption revient à chercher parmi les couvertures celles qui, (d), couvrent le plus d'informations de Q , puis, (e), celles qui contiennent le moins d'informations supplémentaires par rapport à Q. La condition (f) revient à chercher à construire les meilleures couvertures avec le moins de S_i possible (à rest et miss équivalents). Cela permet d'éviter de combiner plusieurs S_i qui ont des descriptions équivalentes.

Comme pour les langages à subsomption structurelle, la découverte des meilleures couvertures dans $\mathcal{ALN}$ est une réécriture qui correspond à l'instance du cadre général (défini à la définition 2.1.1 page 22) qui a les propriétés suivantes :

- $\mathcal{ALN}$ est à la fois le langage source $\mathcal{L}_s$ et de terminologie $\mathcal{L}_t$,
- le langage destination $\mathcal{L}_d$ est $\{\sqcap, \sqcup\}$,
- la relation ρ est la suivante : $Q \rho E \Leftrightarrow (Q \sqcap E \not\equiv \bot$ et $Q - lcs(Q, E) \not\equiv \{Q\})$ et
- on cherche à minimiser par rapport à l'ordre $\preceq$ défini ainsi $E' \preceq E$ si et seulement si $Rest_{E'}(Q) \sqsupseteq Rest_E(Q)$ et si les rest sont équivalents, alors $Miss_{E'}(Q) \sqsupseteq Miss_E(Q)$ et si les miss sont équivalents alors E', en tant qu'ensemble de S_i, est inclus dans E.

Les exemples qui suivent sont destinés à illustrer la notion de meilleure couverture et en particulier la maximalité des rest par rapport à la subsomption ainsi que celle des miss quand les rest sont équivalents. Le problème de construction des couvertures en tant que conjonctions de S_i n'est pas ici abordé, alors que c'est naturellement la principale difficulté de $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$. On part donc d'un ensemble de descriptions candidates à être des meilleures couvertures, et on se contente de les classer, les premières étant les meilleures. De plus, on ne se préoccupe pas ici de la condition (f) imposant de construire les E avec le moins de S_i possibles puisque l'on

suppose les E déjà construites.

L'exemple 17 qui suit est simple dans la mesure où rest et miss sont toujours des descriptions uniques (c'est-à-dire qu'il n'y a pas de cas d'inconsistance implicite générant plusieurs descriptions dans les calculs des rest et miss). L'exemple 18 est plus complexe dans la mesure où l'on traite une situation où les rest et miss peuvent être composés de plusieurs descriptions. Grâce à la notion de subsomption entre ensembles de descriptions, on voit cependant que le raisonnement est très semblable.

Exemple 17

Soient le concept Q et les couvertures E_i suivants :
$Q \equiv Dr\_Info \sqcap Prof \sqcap (\forall membreCP.(\geq 3\ conf)) \sqcap (\forall membreCP.(\leq 6\ conf))$
$E_1 \equiv Dr\_Info \sqcap (\forall membreCP.(\geq 2\ conf)) \sqcap (\geq 21\ publi)$
$E_2 \equiv Dr\_Info \sqcap (\forall membreCP.(\geq 5\ conf)) \sqcap (\geq 25\ publi)$
$E_3 \equiv Dr\_Info \sqcap (\forall membreCP.(\geq 4\ conf))$
$E_4 \equiv Dr\_Maths \sqcap Prof \sqcap (\forall membreCP.(\geq 3\ conf))$
$E_5 \equiv Dr\_Maths \sqcap Prof \sqcap (\forall membreCP.(\geq 4\ conf)$
$E_6 \equiv Dr\_Phys \sqcap Prof \sqcap (\forall membreCP.(\geq 5\ conf)) \sqcap (\geq 26\ publi)$
Leurs rest et miss sont les suivants :

E_i	$Rest_{E_i}(Q)$	$Miss_{E_i}(Q)$
E_1	$Prof \sqcap (\forall membreCP.(\geq 3\ conf)) \sqcap (\forall membreCP.(\leq 6\ conf))$	$(\geq 21\ publi)$
E_2	$Prof \sqcap (\forall membreCP.(\leq 6\ conf))$	$(\geq 25\ publi) \sqcap (\forall membreCP.(\geq 5\ conf))$
E_3	$Prof \sqcap (\forall membreCP.(\leq 6\ conf))$	$(\forall membreCP.(\geq 4\ conf))$
E_4	$Dr\_Info \sqcap (\forall membreCP.(\leq 6\ conf))$	Dr_Maths
E_5	$Dr\_Info \sqcap (\forall membreCP.(\leq 6\ conf))$	$Dr\_Maths \sqcap (\forall membreCP.(\geq 4\ conf))$
E_6	$Dr\_Info \sqcap (\forall membreCP.(\leq 6\ conf))$	$Dr\_Phys \sqcap (\forall membreCP.(\geq 5\ conf)) \sqcap (\geq 26\ publi)$

Les meilleures couvertures sont E_3, E_4 et E_6. En effet,
- d'après la nature des rest, E_2, E_3, E_4, E_5 et E_6 maximisent le rest par rapport à $\sqsubseteq$: on a $Rest_{E_1}(Q) \sqsubseteq Rest_{E_2}(Q)$ et les autres couples de rest sont soit équivalents, soit incomparables par rapport à $\sqsubseteq$,
- et d'après la nature des miss, E_3, E_4 et E_6 maximisent le miss par rapport à $\sqsubseteq$: on a $Miss_{E_2}(Q) \sqsubseteq Miss_{E_3}(Q)$, $Miss_{E_5}(Q) \sqsubseteq Miss_{E_4}(Q)$ et $Miss_{E_6}(Q)$ est incomparable avec $Miss_{E_5}(Q)$ et $Miss_{E_4}(Q)$. Le fait que $Miss_{E_2}(Q) \sqsubseteq Miss_{E_1}(Q)$ et $Miss_{E_6}(Q) \sqsubseteq Miss_{E_2}(Q)$ n'importe pas car le classement sur le rest a déjà départagé E_1, E_2 et E_6.

$\circ$

Exemple 18

Soient Q le concept $\mathcal{ALN}$ à couvrir et E_1 à E_8 des concepts formés comme conjonctions de concepts de $\mathcal{S}_\mathcal{T}$ (i.e. de concepts définis dans $\mathcal{T}$), candidats à être des (meilleures) couvertures de Q en utilisant $\mathcal{T}$.
$Q \equiv Vol \sqcap (\forall escale.\bot) \sqcap (\forall lieuArr.(\geq 2\ mag)) \sqcap (\forall lieuArr.AeroInt) \sqcap (\forall reduc.CadreSup) \sqcap (\forall reduc.(\geq 2\ enf))$
$E_1 \equiv Croisiere \sqcap (\forall escale.Port) \sqcap (\forall escale.(\leq 10000\ hab)) \sqcap (\forall lieuArr.(\geq 1\ mag)) \sqcap (\forall reduc.CadreSup)$
$E_2 \equiv Vol \sqcap (\forall lieuArr.(\geq 3\ mag))$
$E_3 \equiv Voyage \sqcap (\forall escale.\bot) \sqcap (\forall lieuArr.(\geq 1\ mag)) \sqcap (\forall reduc.CadreSup) \sqcap (\forall reduc.(\geq 4\ enf)) \sqcap (\forall reduc.(\leq 6\ enf))$

$E_4 \equiv Voyage \sqcap (\forall escale.\bot) \sqcap (\forall reduc.\bot)$
$E_5 \equiv Croisiere \sqcap (\forall lieuArr.VilleUSA) \sqcap (\forall reduc.\neg CadreSup)$
$E_6 \equiv Croisiere \sqcap (\forall escale.VilleUE) \sqcap (\forall escale.SiteArcheo) \sqcap (\forall escale.(\leq 5000\ hab)) \sqcap (\forall reduc.\bot)$
$E_7 \equiv Vol \sqcap (\forall escale.\bot) \sqcap (\forall lieuArr.(\geq 4\ mag)) \sqcap (\forall lieuArr.AeroInt) \sqcap (\forall reduc.\bot)$
$E_8 \equiv Voyage \sqcap (\forall escale.\bot) \sqcap (\forall reduc.\bot) \sqcap (\forall classe.Premiere)$
Leurs rest et miss sont les suivants :

E_i	$Rest_{E_i}(Q)$
E_1	$Vol \sqcap (\forall escale.\neg Port) \sqcap (\forall lieuArr.(\geq 2\ mag)) \sqcap (\forall lieuArr.AeroInt) \sqcap (\forall reduc.(\geq 2\ enf))$ $Vol \sqcap (\forall escale.(\geq 10001\ hab)) \sqcap (\forall lieuArr.(\geq 2\ mag)) \sqcap (\forall lieuArr.AeroInt) \sqcap (\forall reduc.(\geq 2\ enf))$
E_2	$(\forall escale.\bot) \sqcap (\forall lieuArr.AeroInt) \sqcap (\forall reduc.CadreSup) \sqcap (\forall reduc.(\geq 2\ enf))$
E_3	$Vol \sqcap (\forall lieuArr.(\geq 2\ mag)) \sqcap (\forall lieuArr.AeroInt)$
E_4	$Vol \sqcap (\forall lieuArr.(\geq 2\ mag)) \sqcap (\forall lieuArr.AeroInt)$
E_5	Q
E_6	$Vol \sqcap (\forall escale.\neg VilleUE) \sqcap (\forall lieuArr.(\geq 2\ mag)) \sqcap (\forall lieuArr.AeroInt)$ $Vol \sqcap (\forall escale.\neg SiteArcheo) \sqcap (\forall lieuArr.(\geq 2\ mag)) \sqcap (\forall lieuArr.AeroInt)$ $Vol \sqcap (\forall escale.(\geq 5001\ hab)) \sqcap (\forall lieuArr.(\geq 2\ mag)) \sqcap (\forall lieuArr.AeroInt)$
E_7	$\top$
E_8	$Vol \sqcap (\forall lieuArr.(\geq 2\ mag)) \sqcap (\forall lieuArr.AeroInt)$

E_5 n'est pas une couverture, puisque $Rest_{E_5}(Q) \equiv Q$.
Concernant les rest, on a les relations suivantes : $Rest_{E_3}(Q) \equiv Rest_{E_4}(Q) \equiv Rest_{E_8}(Q)$, $Rest_{E_1}(Q) \sqsubseteq Rest_{E_3}(Q)$, $Rest_{E_6}(Q) \sqsubseteq Rest_{E_3}(Q)$, $Rest_{E_3}(Q) \sqsubseteq Rest_{E_7}(Q)$ et $Rest_{E_2}(Q) \sqsubseteq Rest_{E_7}(Q)$. Les autres couples de rest sont incomparables.

E_i	$Miss_{E_i}(Q)$
E_1	$Croisiere$
E_2	$\forall lieuArr.(\geq 3\ mag)$
E_3	$Voyage \sqcap (reduction.(\geq 4\ enf)) \sqcap (reduction.(\leq 6\ enf))$
E_4	$Voyage \sqcap (\forall reduction.\neg CadreSup)$ $Voyage \sqcap (\forall reduction.(\leq 1\ enf))$
E_5	E_5
E_6	$Croisiere \sqcap (\forall reduction.\neg CadreSup)$ $Croisiere \sqcap (\forall reduction.(\leq 1\ enf))$
E_7	$(\forall lieuArr.(\geq 4\ mag)) \sqcap (\forall reduction.\neg CadreSup)$ $(\forall lieuArr.(\geq 4\ mag)) \sqcap (\forall reduction.(\leq 1\ enf))$
E_8	$Voyage \sqcap (\forall reduction.\neg CadreSup) \sqcap (\forall classe.Premiere)$ $Voyage \sqcap (\forall reduction.(\leq 1\ enf)) \sqcap (\forall classe.Premiere)$

Concernant les miss, on a les relations suivantes : $Miss_{E_6}(Q) \sqsubseteq Miss_{E_1}(Q)$, $Miss_{E_7}(Q) \sqsubseteq Miss_{E_2}(Q)$ et $Miss_{E_8}(Q) \sqsubseteq Miss_{E_4}(Q)$. Les autres couples de miss sont incomparables. On a donc :

Classement par rapport au rest uniquement (du plus grand au plus petit par rapport à la subsomption)	Meilleures couvertures de Q (en fonction du rest puis du miss)
1. E_7 2. E_2, E_3, E_4 et E_8 3. E_1 et E_6	1. E_7 2. E_2, E_3 et E_4 3. E_8 4. E_1 et E_6

On remarque que E_1 et E_6 sont classés au même niveau : cela provient du fait que E_1 et E_6 ont des rest incomparables et que l'on classe en fonction du miss pour des E_i de rest équivalents. Ainsi le fait que $Miss_{E_6}(Q) \sqsubseteq Miss_{E_1}(Q)$ n'est pas exploité ici. ◦

Dans les sections 4.3 et 4.4, on détaille la construction des meilleures couvertures comme conjonctions de concepts définis dans une terminologie $\mathcal{T}$.

4.3 Etude du problème $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$

Comme cela est évoqué dans la section 4.1, nous avons choisi de chercher les meilleures couvertures dans $\mathcal{ALN}$ en réduisant petit à petit l'espace de recherche qui est le treillis des parties de $\mathcal{S}_\mathcal{T}$. Pour ce faire, nous cherchons comment chaque condition de (b) à (f) de la définition des meilleures couvertures[16] peut permettre le calcul d'une borne supérieure ou inférieure de l'ensemble des meilleures couvertures. Par borne, nous entendons ici un ensemble de parties de $\mathcal{S}_\mathcal{T}$ tel que chaque meilleure couverture est contenue dans au moins une des parties (cas d'une borne supérieure), ou contient au moins une des parties (cas d'une borne inférieure). Ainsi, la démarche est la suivante :

Section 4.3.1 : La condition (b), caractérisée par le lemme 4.3.1 page 60, implique de réduire l'espace de recherche en supprimant tous les ensembles de S_i dont la conjonction avec Q est inconsistante. Ceci revient à chercher la borne supérieure contenant les plus grands ensembles de S_i dont la conjonction avec Q est consistante (plus grands au sens inclusion). On nomme S_{cons} cette première borne supérieure des meilleures couvertures.

Section 4.3.2 : La condition (c), caractérisée par le lemme 4.3.2 page 63, implique une borne inférieure qui contient tous les plus petits ensembles de S_i ayant une partie en commun avec Q. On nomme I_{couv} cette première borne inférieure. Grâce à cette borne, on peut affiner S_{cons} en une seconde borne supérieure nommée S_{couv} regroupant les plus grands ensembles de S_i consistants avec Q et ayant une partie en commun avec Q.

Section 4.3.3 : La condition (d), caractérisée par le lemme 4.3.3 page 65 et les lemmes 4.3.4 et 4.3.5 page 66 qui en découlent, permet d'isoler parmi les plus grandes couvertures, éléments de S_{couv}, celles qui ont un rest maximal. Ces dernières sont regroupées dans un ensemble nommé S_{rest}. S_{rest} est donc la troisième borne supérieure contenant les plus grandes couvertures de rest maximal (grandes par rapport à l'inclusion, et maximal par rapport à la subsomption). La condition (d) permet de plus d'obtenir l'ensemble des plus petites couvertures de rest max. On nomme I_{rest} cette deuxième borne inférieure. Elle est obtenue de la manière suivante. A partir de chaque élément E^* de S_{rest}, on note par $R_{eq}(E^*)$ l'ensemble des plus petits sous-ensembles de E^* qui ont un rest équivalent à E^*. I_{rest} est l'union des ces ensembles $R_{eq}(E^*)$.

[16] La condition (a) (qui définit les meilleures couvertures comme des conjonctions de S_i) définit quant à elle le fait que l'espace de recherche soit le treillis des parties de $\mathcal{S}_\mathcal{T}$.

Nature de la borne	Nom	Contenu
Borne supérieure 1	S_{cons}	Tous les plus grands ensembles de S_i consistants avec Q (par conjonction).
Borne inférieure 1	I_{couv}	Tous les plus petits ensembles de S_i consistants avec Q et ayant au moins une partie en commun avec Q.
Borne supérieure 2	S_{couv}	Toutes les plus grandes couvertures de Q.
Borne supérieure 3	S_{rest}	Toutes les plus grandes couvertures de Q de rest maximal.
Borne inférieure 2	I_{rest}	Toutes les plus petites couvertures de Q de rest maximal.

TAB. 4.2 – Les bornes supérieures et inférieures de l'espace de recherche calculées au cours de la découverte des meilleures couvertures dans $\mathcal{ALN}$.

Section 4.3.4 : La condition (e) apparaît plus complexe que les précédentes. Aucune caractérisation n'ayant été trouvée, elle se résoud par un parcours des solutions comprises entre les deux bornes précédentes. La condition (f) enfin se résoud par un tri par rapport à l'inclusion des ensembles de S_i résultats des étapes précédentes.

Le tableau 4.2 récapitule les bornes qui viennent d'être présentées. De même, le tableau 4.9, page 86, constitue un lexique pour les noms des ensembles utilisés dans ce chapitre. Il est utile pour cette section et la section suivante 4.4.

Pour illustrer les réductions successives de l'espace de recherche grâce aux bornes supérieures et inférieures, nous présentons maintenant l'exemple "courant". Cet exemple sera repris dans la section 4.4 dans le but d'illustrer les mécanismes de calcul des bornes.

Il est important de remarquer que les caractérisations proposées dans la suite sont exprimées en termes de clauses. Par conséquent, ceci permet d'éviter la normalisation de conjonctions de S_i, le calcul direct de différences sémantiques ainsi que les tests de subsomption entre des ensembles de descriptions. On effectue ces tâches en ne faisant que comparer des clauses entre elles, ce qui est beaucoup moins coûteux. Un second avantage de travailler sur des clauses est qu'on n'effectue les tâches qu'une seule fois. Par exemple, les cas d'inconsistance explicite dans les conjonctions de S_i ne seront découverts qu'une fois si on découvre les inconsistances explicites entre clauses qui sont à leur origine. Travailler au niveau des conjonctions de S_i nécessiterait de devoir découvrir certaines inconsistances plusieurs fois (i.e. dans plusieurs conjonctions de S_i) alors qu'elles proviennent peut-être d'une seule et même inconsistance dans un seul ensemble de clauses. Travailler au niveau des clauses est donc moins coûteux, et justifie l'usage de la forme normale à base de clauses.

Exemple courant 1

Nous supposons que nous recherchons les $\mathcal{ALN}$-meilleures couvertures de Q en utilisant $\mathcal{T}$, avec $\mathcal{T}$ et Q définies dans $\mathcal{ALN}$ et données dans le tableau 4.3. Les concepts de la terminologie décrivent trois types de trajet qui sont tous des vols avec des informations sur le lieu d'arrivée, les éventuelles escales et les animaux acceptés pendant le vol, quatre types d'hôtel comme des hébergements avec des informations sur le nombre d'étoiles qui leur sont attribuées, d'éventuels équipements de loisirs, le type de télévision disponible et les animaux acceptés, ainsi qu'un type de voiture (les berlines 4x4). Q décrit un hébergement et un vol avec leurs caractéristiques. Notons que les S_i sont ici notés T_1 à T_3, H_1 à H_4 et V_1 pour permettre leur identification plus facilement (Trajet, Hôtel ou Voiture).

La figure 4.1 montre le treillis des parties de $\mathcal{S}_\mathcal{T}$ qui est l'espace de recherche des meilleures couvertures. Comme $|\mathcal{S}_\mathcal{T}| = 8$, il y a au total $2^8 = 256$ ensembles de S_i possibles. Les bornes

$Trajet : T_1 \equiv$	$Vol \sqcap \forall lieuArr.Ville \sqcap \forall escale.\forall duree. \leq 2heure \sqcap$ $\forall escale.\forall mag. \geq 1DutyFree \sqcap \forall escale. \geq 1mag \sqcap \forall aniAccept.\forall transport.Cage$
$Trajet : T_2 \equiv$	$Vol \sqcap \forall lieuArr.Ville \sqcap \forall escale.\forall mag.BoutiqueLuxe \sqcap$ $\forall escale.\forall mag. \leq 0DutyFree \sqcap \forall aniAccept.Chien \sqcap \forall tv.EcranPlat$
$Trajet : T_3 \equiv$	$Vol \sqcap \forall lieuArr.Ville \sqcap \forall lieuArr.\forall situe.France \sqcap \forall escale.\forall duree. \leq 1heure \sqcap$ $\geq 1escale \sqcap \forall aniAccept. \leq 10kg$
$Hotel : H_1 \equiv$	$Heberg \sqcap \leq 2etoile \sqcap \geq 2etoile \sqcap \forall equip.Piscine \sqcap \forall tv.\forall chaine.Satellite \sqcap$ $\forall tv. \geq 10chaine \sqcap \forall equip.Tennis \sqcap \forall aniAccept.\forall transport.Cage$
$Hotel : H_2 \equiv$	$Heberg \sqcap \geq 1etoile \sqcap \forall tv.\forall chaine.\neg Satellite \sqcap \forall tv.EcranPlat \sqcap$ $\forall aniAccept. \leq 10kg \sqcap \forall aniAccept.Chien$
$Hotel : H_3 \equiv$	$Heberg \sqcap \forall tv.\bot \sqcap \leq 1etoile$
$Hotel : H_4 \equiv$	$Heberg \sqcap \leq 3etoile \sqcap \geq 3etoile \sqcap \forall equip.Piscine$
$Voiture : V_1 \equiv$	$Vehicule \sqcap \forall categorie.Berline \sqcap \forall motorisation.4x4$

$Q \equiv$	$Vol \sqcap \forall lieuArr.Ville \sqcap \forall lieuArr. \leq 100000hab \sqcap \forall escale.\forall duree. \leq 1heure \sqcap \forall aniAccept.\bot \sqcap$ $Heberg \sqcap \leq 2etoile \sqcap \forall equip.Piscine \sqcap \forall tv.EcranPlat$

Tab. 4.3 – La terminologie $\mathcal{T}$ et le concept Q à couvrir dans l'exemple courant.

supérieure et inférieure par défaut sont indiquées en gras dans la figure et sont respectivement l'ensemble $\mathcal{S}_\mathcal{T}$ et l'ensemble vide. ◦

Nous examinons maintenant les conditions (b) à (f) de la définition des meilleures couvertures. Pour les trois premières, nous donnons une caractérisation aboutissant à une réduction de l'espace de recherche au travers de la définition d'une borne supérieure ou inférieure. Nous précisons ensuite comment les conditions (e) et (f) sont vérifiées. Nous illustrons chaque étape sur l'exemple courant. Nous rappelons que cette section 4.3 présente les résultats théoriques qui permettent une approche par réductions successives de l'espace de recherche. Toutes les méthodes de calcul des bornes présentées ici seront détaillées à la section 4.4.

4.3.1 Caractérisation de la condition (b) de la définition 4.2.5

Selon la définition 4.2.5, le premier critère définissant une couverture E de Q est sa consistance avec Q par conjonction (condition (b)). Il est clair que cette condition est vérifiée si et seulement s'il n'existe pas dans l'ensemble des clauses des S_i de E et de Q (après normalisation) de sous-ensemble de clauses dont la conjonction est inconsistante. C'est précisément ce que formalise le lemme 4.3.1.

Lemme 4.3.1
Soit $\mathcal{T}$ une $\mathcal{ALN}$-terminologie composée uniquement de définitions de concepts. Soit $\mathcal{S}_\mathcal{T}$ l'ensemble des concepts définis S_i de $\mathcal{T}$ et $\mathcal{S}_{\mathcal{T}_Q} = \mathcal{S}_\mathcal{T} \cup \{Q\}$. Soit E une conjonction de S_i et Q une $\mathcal{ALN}$-description consistante ($Q \not\equiv \bot$). On a :

$$E \sqcap Q \not\equiv \bot \Leftrightarrow \forall P \subseteq \left(\left(\bigcup_{S_i \in E} \widehat{\mathcal{G}}^{\#}_{S_i} \right) \cup \widehat{\mathcal{G}}^{\#}_{Q} \right), \sqcap_{c \in P} c \not\equiv \bot$$

Ce lemme implique qu'il faut chercher tous les ensembles de clauses, parmi les clauses des S_i et de Q, dont la conjonction est inconsistante. Sans perte de généralité, on peut se limiter aux plus petits de ces ensembles par inclusion. On déduit alors de ces ensembles les plus petits ensembles de S_i dont la conjonction avec Q est inconsistante. L'ensemble de ces ensembles de S_i est appelé *Inc*. Comme on cherche des ensembles de S_i consistants avec Q par conjonction, on

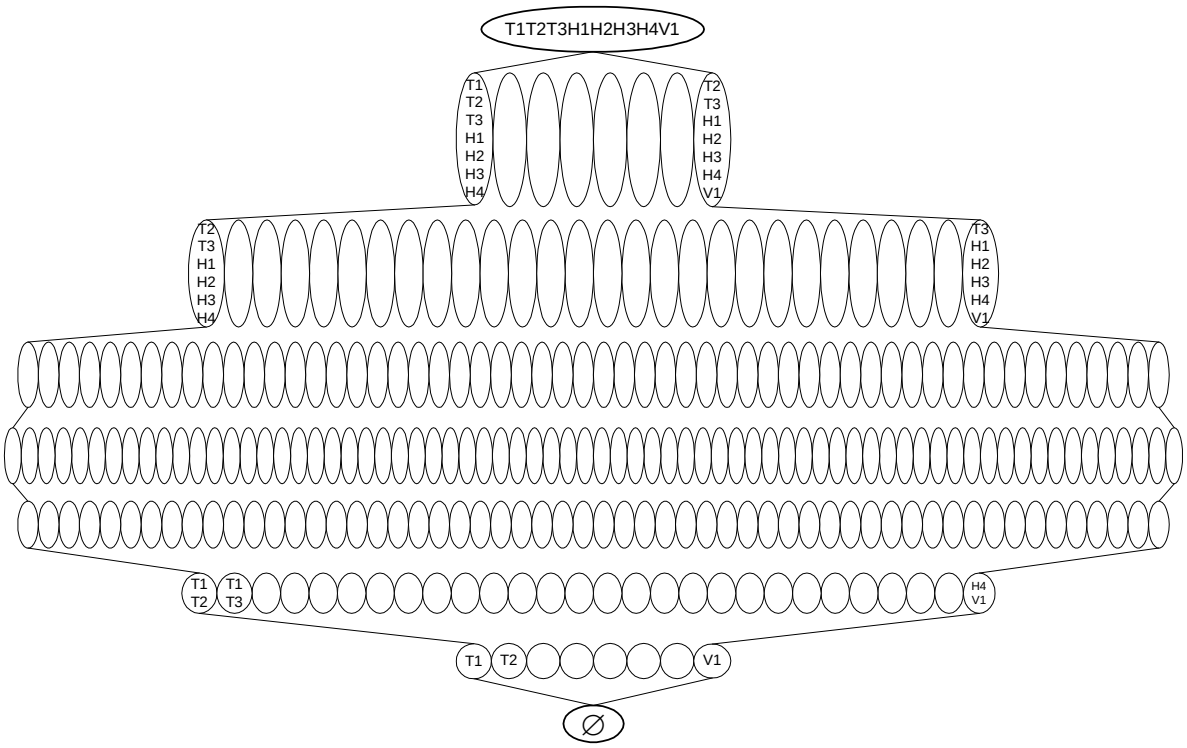


FIG. 4.1 – L'espace de recherche initial des meilleures couvertures de l'exemple courant.

va oter de l'espace de recherche tous les ensembles de Inc et tous leurs sur-ensembles. Cela se traduit par le calcul d'une borne supérieure contenant tous les plus grands ensembles de S_i dont la conjonction avec Q est consistante. Comme on le verra dans la section 4.4, ces plus grands ensembles consistants peuvent être calculés à partir de Inc. Ils sont regroupés dans l'ensemble S_{cons} qui est donc la première borne supérieure limitant l'espace de recherche.
$S_{cons} = \{E \subseteq \mathcal{S}_\mathcal{T} \mid E \sqcap Q \not\equiv \bot$ et E maximal par rapport à l'inclusion$\}$

Exemple courant 2
Les plus petites conjonctions de clauses de S_i inconsistantes, ou inconsistantes avec une ou plusieurs clauses de Q sont :
– $\forall escale.\forall mag. \geq 1 DutyFree \sqcap \forall escale.\forall mag. \leq 0 DutyFree \sqcap \forall escale. \geq 1 mag \sqcap \geq 1 escale \equiv \bot$, qui apparaît dans $T_1 \sqcap T_2 \sqcap T_3$,
– $\leq 2 etoile \sqcap \geq 3 etoile \equiv \bot$, qui apparaît dans $H_4 \sqcap Q$ et dans $H_1 \sqcap H_4$,
– $\leq 1 etoile \sqcap \geq 3 etoile \equiv \bot$, qui apparaît dans $H_3 \sqcap H_4$ et
– $\leq 1 etoile \sqcap \geq 2 etoile \equiv \bot$, qui apparaît dans $H_1 \sqcap H_3$.

Ainsi, $Inc = \{\{T_1, T_2, T_3\}, \{H_4\}, \{H_1, H_3\}\}$. Ceci permet de trouver que les plus grands ensembles de S_i dont la conjonction avec Q est consistante sont :
$S_{cons} = \{E_1 = \{T_2, T_3, H_2, H_3, V_1\}, E_2 = \{T_2, T_3, H_1, H_2, V_1\}, E_3 = \{T_1, T_3, H_2, H_3, V_1\},$
$E_4 = \{T_1, T_3, H_1, H_2, V_1\}, E_5 = \{T_1, T_2, H_2, H_3, V_1\}, E_6 = \{T_1, T_2, H_1, H_2, V_1\}\}$
S_{cons} est donc la première borne supérieure. La figure 4.2 montre l'espace de recherche limité par S_{cons}. Les éléments en grisé sont les parties de l'espace de recherche élaguées par la borne S_{cons}. On peut observer qu'il ne reste plus que 6 solutions de cardinalité 5, 21 de cardinalité 4, 29 de cardinalité 3, 20 de cardinalité 2, 7 de cardinalité 1 et une de cardinalité 0. Soit 84 ensembles

de S_i possibles. ◦

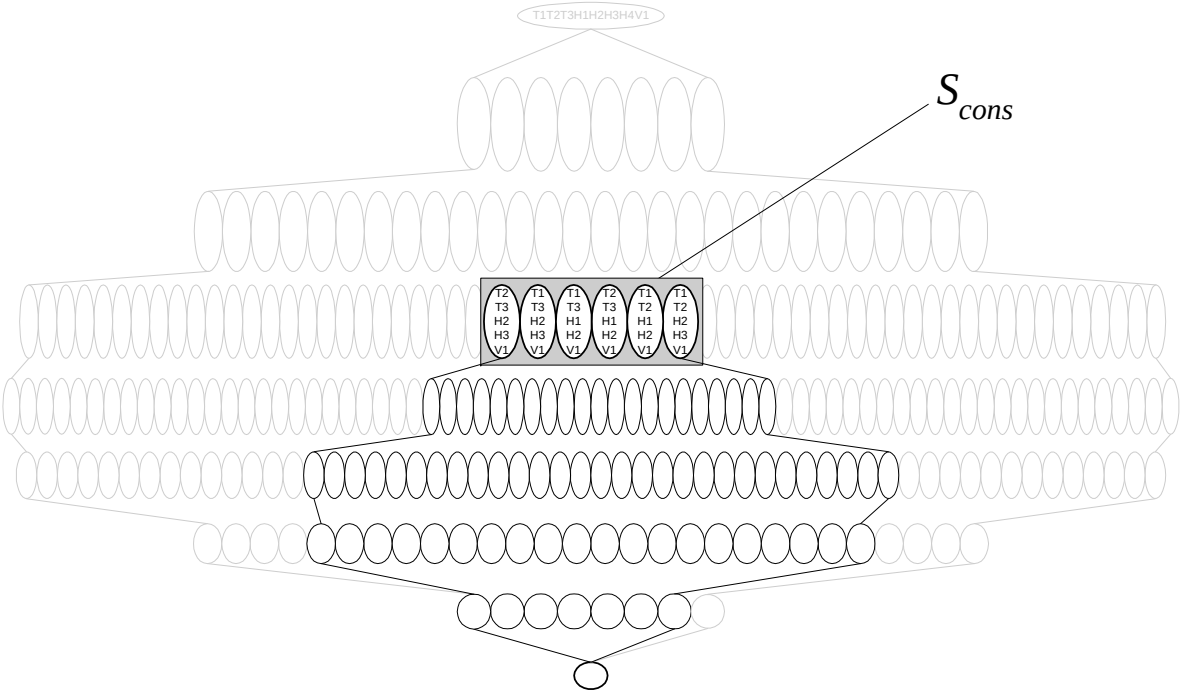


FIG. 4.2 – L'espace de recherche des meilleures couvertures de l'exemple courant limité par la première borne supérieure S_{cons}.

4.3.2 Caractérisation de la condition (c) de la définition 4.2.5

Selon la définition 4.2.5, le second critère définissant une couverture de Q est le fait qu'elle partage des informations avec Q (condition (c)). Le lemme 4.3.2 caractérise cette propriété en disant qu'elle est équivalente à l'existence d'une clause commune entre Q et chaque couverture E. Pour définir ce qu'est une clause en commun, nous avons besoin de définir la notion de couverture d'une clause.

Définition 4.3.1 (Couverture d'une clause)
Soient Q et E deux $\mathcal{ALN}$-descriptions consistantes, c_Q une clause de Q (i.e. $c_Q \in \widehat{\mathcal{G}}_Q^{\#}$) et c_E une clause de E (i.e. $c_E \in \widehat{\mathcal{G}}_E^{\#}$). On définit :

- *c_Q est directement couverte par c_E ssi $c_E \sqsubseteq c_Q$.*
- *c_Q est indirectement couverte par c_E ssi*
  $c_Q = \forall R_1 R_2 ... R_{n-1}.(\leq 0\ R_n)$, $n \geq 1$ et $c_E = \forall R_1 R_2 ... R_{n+m}.P$, $m \geq 0$ où P est un concept atomique, une négation de concept atomique ou une restriction numérique (et $P \neq \bot$).
  (Autrement dit, c_Q est indirectement couverte par c_E ssi $c_Q \sqsubseteq c_E$ et $prof(c_Q) < prof(c_E)$.)
- *c_Q est couverte par c_E ssi c_Q est directement couverte par c_E ou c_Q est indirectement couverte par c_E.*
- *c_Q est directement couverte par E ssi $\exists c_E \in \widehat{\mathcal{G}}_E^{\#} \mid c_Q$ est directement couverte par c_E.*
- *c_Q est indirectement couverte par E ssi $\exists c_E \in \widehat{\mathcal{G}}_E^{\#} \mid c_Q$ est indirectement couverte par c_E.*

– *c_Q est couverte par E ssi c_Q est directement couverte par E ou c_Q est indirectement couverte par E*

On peut maintenant formuler le lemme qui caractérise la condition (c) de la définition 4.2.5. Celui-ci dit que Q a une clause en commun avec une conjonction E de S_i si et seulement si Q possède au moins une clause couverte par E (après normalisation).

Lemme 4.3.2
Soient Q et E deux $\mathcal{ALN}$-descriptions de concepts consistantes. On a :

$$Q - lcs(Q, E) \not\equiv \{Q\} \Leftrightarrow \exists c_Q \in \widehat{\mathcal{G}}^{\#}_Q \mid c_Q \text{ est couverte par } E$$

Ce lemme découle directement des caractérisations de la subsomption structurelle (théorème 4.2.1 page 49), du lcs (lemme 4.2.1 page 50) et de la différence sémantique (théorème 4.2.2 page 52). Sa démonstration détaillée est donnée dans l'annexe A.5 page 153.

Ce lemme implique qu'il faut enlever de l'espace de recherche les conjonctions de S_i qui, après normalisation, ne possèdent aucune clause couvrant (directement ou indirectement) une clause de Q. Pour ce faire, il faut connaître toutes les clauses des conjonctions de S_i normalisées de l'espace de recherche. De par la nature des règles de normalisation, il est simple de constater qu'une clause appartenant à une conjonction de S_i normalisée (et $\not\equiv \bot$) provient soit de la forme normalisée d'un des S_i, soit d'une inconsistance implicite apparaissant dans un sous-ensemble de S_i (c'est alors une clause d'exclusion). Sans perte de généralité, on peut considérer que ce sous-ensemble de S_i est minimal par rapport à l'inclusion. En considérant que l'on peut découvrir toutes les clauses d'exclusion (i.e. tous les cas d'inconsistance implicite), on a, avec les clauses des S_i normalisés, toutes les clauses possibles pouvant appartenir à une conjonction normalisée de S_i. On peut alors trouver tous les cas où une clause c_Q de Q est couverte par un ensemble de S_i normalisé. On décrits tous ces cas dans des triplets (c, H, c_Q) où H est un ensemble de S_i minimal par rapport à l'inclusion, c est une clause de H normalisé, c_Q est une clause de Q, et c_Q est couverte par c. Les triplets (c, H, c_Q) sont regroupés dans deux ensembles nommés C_{dir} et C_{indir}. C_{dir} recense les couvertures directes des clauses de Q, et C_{indir} les couvertures indirectes.

$$\begin{aligned} C_{dir} = \{(c, H, c_Q) \mid\ & H \subseteq \mathcal{S}_\mathcal{T} \text{ et } H \text{ minimal par rapport à } \subseteq \text{ et} \\ & c \in \widehat{\mathcal{G}}^{\#}_{\sqcap_{S_i \in H} S_i} \text{ et } c_Q \in \widehat{\mathcal{G}}^{\#}_Q \text{ et} \\ & c \text{ couvre directement } c_Q\} \end{aligned}$$

$$\begin{aligned} C_{indir} = \{(c, H, c_Q) \mid\ & H \subseteq \mathcal{S}_\mathcal{T} \text{ et } H \text{ minimal par rapport à } \subseteq \text{ et} \\ & c \in \widehat{\mathcal{G}}^{\#}_{\sqcap_{S_i \in H} S_i} \text{ et } c_Q \in \widehat{\mathcal{G}}^{\#}_Q \text{ et} \\ & c \text{ couvre indirectement } c_Q\} \end{aligned}$$

D'après le lemme 4.3.2, toute couverture de Q doit couvrir au moins une clause de Q. Comme C_{dir} et C_{indir} regroupent tous les plus petits ensembles H de S_i couvrant une clause de Q, alors toute couverture de Q est sur-ensemble d'au moins un H de C_{dir} ou C_{indir}. On déduit donc de C_{dir} et C_{indir} une borne inférieure de l'espace de recherche. On nomme I_{couv} cette première borne inférieure.

I_{couv} est constituée d'un sous-ensemble de l'ensemble des H de $C_{dir} \cup C_{indir}$: seuls les H qui sont sous-ensembles d'un élément de la borne supérieure S_{cons} (et sont donc consistants par conjonction avec Q) font partie de I_{couv}. Réciproquement, on peut oter de S_{cons} les ensembles de S_i qui ne sont sur-ensembles d'aucun H de $C_{dir} \cup C_{indir}$, puisqu'alors on sait que ces ensembles ne couvrent aucune clause de Q. Cela vient du fait que l'ajout par conjonction d'un S_j à H conserve la propriété du lemme 4.3.2. On nomme S_{couv} la nouvelle borne supérieure obtenue.

En résumé :
$I_{couv} = \{H \mid \exists(c, H, c_Q) \in C_{dir} \cup C_{indir}$ et H est sous-ensemble d'un ensemble E de S_i de $S_{cons}\}$
$S_{couv} = \{E \in S_{cons} \mid E$ est sur-ensemble d'un ensemble H de S_i de $I_{couv}\}$

Le théorème 4.3.1 résume ce qui vient d'être dit en donnant une caractérisation des couvertures d'une $\mathcal{ALN}$-description en utilisant une $\mathcal{ALN}$-terminologie.

Théorème 4.3.1 (Caractérisation des couvertures $\mathcal{ALN}$)
Soit Q une $\mathcal{ALN}$-description. Soit $\mathcal{T}$ une $\mathcal{ALN}$-terminologie avec $\mathcal{S}_\mathcal{T}$ l'ensemble des concepts définis S_i. Soient S_{couv} et I_{couv} les ensembles définis précédemment.

Un ensemble E_{couv} de S_i (i.e. $E_{couv} \subseteq \mathcal{S}_\mathcal{T}$) est une couverture $\mathcal{ALN}$ de Q en utilisant $\mathcal{T}$ si et seulement si :

- *E_{couv} est un sous-ensemble d'un élément E' de S_{couv} et*
- *E_{couv} est un sur-ensemble d'un élément H de I_{couv}*

Exemple courant 3
Les cas d'inconsistance implicite sont au nombre de 2 :
– Dans $T_1 \sqcap T_2$ on trouve la conjonction $\forall escale.\forall mag.(\geq 1 DutyFree) \sqcap \forall escale.\forall mag.(\leq 0 DutyFree) \sqcap \forall escale.(\geq 1 mag)$ qui est normalisée en la clause d'exclusion $\forall escale.\bot \equiv \leq 0 escale$.
– Dans $H_1 \sqcap H_2$ on trouve $\forall tv.\forall chaine.Satellite \sqcap \forall tv. \geq 10 chaine \sqcap \forall tv.\forall chaine.\neg Satellite$ qui est normalisée en la clause d'exclusion $\forall tv.\bot \equiv \leq 0 tv$.
A ce stade, on connaît toutes les clauses des S_i normalisés. On vient de découvrir toutes les clauses d'exclusion et les plus petits ensembles de S_i qui les contenaient. En comparant toutes ces clauses deux à deux avec les clauses de Q, on obtient les ensembles C_{dir} et C_{indir} donnés au tableau 4.4. Les clauses d'exclusion ou des S_i couvrant directement (respectivement indirectement) une clause de Q induisent les triplets de C_{dir} (respectivement de C_{indir}).
En examinant C_{dir} et C_{indir}, on peut voir que les plus petits ensembles de S_i ayant une clause en commun avec Q sont les singletons $\{T_1\}$, $\{T_2\}$, $\{T_3\}$, $\{H_1\}$, $\{H_2\}$, $\{H_3\}$ et $\{H_4\}$ (pas $\{V_1\}$). I_{couv} est donc constituée de plusieurs de ces singletons.
Comme H_4 n'est sous-ensemble d'aucun élément de S_{cons}, H_4 n'appartient pas à I_{couv} car aucun sur-ensemble de $\{H_4\}$ ne pourra être sous-ensemble d'un élément de S_{cons}. Les autres singletons constituent I_{couv}, la première borne inférieure de l'espace de recherche. I_{couv} n'étant pas vide, l'ensemble vide ne fait plus partie des solutions possibles. De même, il faut oter de S_{cons} tous les ensembles de S_i qui ne contiennent pas au moins un élément de cette borne inférieure, puisque ces ensembles et leurs sous-ensembles ne couvrent aucune clause de Q. Ici, tous les éléments de S_{cons} contiennent au moins un singleton de I_{couv}. Donc $S_{couv} = S_{cons}$.
S_{couv} et I_{couv} délimitent ainsi les couvertures de Q en utilisant $\mathcal{T}$. La figure 4.3 montre l'ensemble de ces couvertures.

○

4.3.3 Caractérisation de la condition (d) de la définition 4.2.6

Selon la définition 4.2.6, la condition (d) de la définition des meilleures couvertures est le fait d'avoir un rest maximal par rapport à la subsomption (entre ensembles de descriptions). Le lemme 4.3.3 donne une caractérisation en termes de clauses de la subsomption entre deux rest. Ce lemme dit qu'une couverture E_1 a un rest qui subsume celui d'une couverture E_2 si et seulement si toute clause de Q couverte par E_2 est aussi couverte par E_1 et si toute clause de

$C_{dir} =$	$\{(Vol, T_1, Vol)$,
	(Vol, T_2, Vol),
	(Vol, T_3, Vol),
	$(\forall lieuArr.Ville, T_1, \forall lieuArr.Ville)$,
	$(\forall lieuArr.Ville, T_2, \forall lieuArr.Ville)$,
	$(\forall lieuArr.Ville, T_3, \forall lieuArr.Ville)$,
	$(Heberg, H_1, Heberg)$,
	$(Heberg, H_2, Heberg)$,
	$(Heberg, H_3, Heberg)$,
	$(Heberg, H_4, Heberg)$,
	$(\leq 2etoile, H_1, \leq 2etoile)$,
	$(\leq 1etoile, H_3, \leq 2etoile)$,
	$(\forall equip.Piscine, H_1, \forall equip.Piscine)$,
	$(\forall equip.Piscine, H_4, \forall equip.Piscine)$,
	$(\forall tv.EcranPlat, T_2, \forall tv.EcranPlat)$,
	$(\forall tv.EcranPlat, H_2, \forall tv.EcranPlat)$,
	$(\leq 0tv, H_3, \forall tv.EcranPlat)$,
	$(\forall escale.\forall duree. \leq 1heure, T_3, \forall escale.\forall duree. \leq 1heure)$,
	$(\leq 0escale, T_1T_2, \forall escale.\forall duree. \leq 1heure)$,
	$(\leq 0tv, H_1H_2, \forall tv.EcranPlat)\}$
$C_{indir} =$	$\{(\forall aniAccept.\forall transport.Cage, H_1, \leq 0aniAccept)$,
	$(\forall aniAccept.\forall transport.Cage, T_1, \leq 0aniAccept)$,
	$(\forall aniAccept.Chien, H_2, \leq 0aniAccept)$,
	$(\forall aniAccept.Chien, T_2, \leq 0aniAccept)$,
	$(\forall aniAccept. \leq 10kg, H_2, \leq 0aniAccept)$,
	$(\forall aniAccept. \leq 10kg, T_3, \leq 0aniAccept)\}$

TAB. 4.4 – Les ensembles C_{dir} et C_{indir} permettant d'obtenir la première borne inférieure nommée I_{couv} de l'espace de recherche dans l'exemple courant.

Q couverte indirectement par E_1 ne peut l'être qu'indirectement par E_2 et par des clauses qui subsument E_1.

Lemme 4.3.3
Soit Q une $\mathcal{ALN}$-description. Soit $\mathcal{T}$ une $\mathcal{ALN}$-terminologie avec $\mathcal{S}_\mathcal{T}$ l'ensemble des concepts définis S_i. Soient E_1 et E_2 deux couvertures de Q en utilisant $\mathcal{T}$. On a :

$Rest_{E_2}(Q) \sqsubseteq Rest_{E_1}(Q) \quad \Leftrightarrow \quad \forall c_Q \in \widehat{\mathcal{G}}^{\#}_Q$,
- *si c_Q est couverte par E_2, alors c_Q est couverte par E_1, et*
- *si c_Q est indirectement couverte par E_1*
  alors
  c_Q n'est pas directement couverte par E_2, et
  $\forall c_{E_2} \in \widehat{\mathcal{G}}^{\#}_{E_2} \mid c_{E_2}$ couvre indirectement c_Q, on a $E_1 \sqsubseteq c_{E_2}$

La démonstration de ce lemme est donnée dans l'annexe A.6 page 154. Son principe est de démontrer la condition nécessaire par sa contraposée en effectuant une étude de cas sur toutes les configurations de clauses de Q possibles. La condition suffisante est démontrée directement

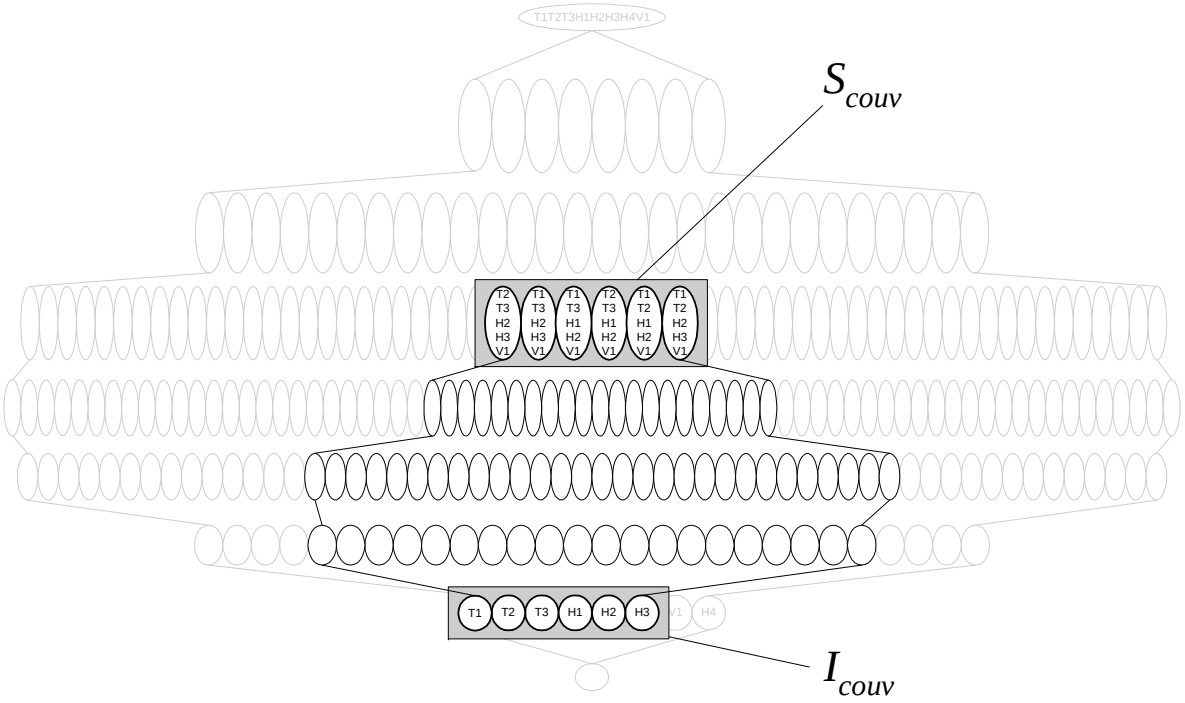


FIG. 4.3 – L'espace de recherche des meilleures couvertures de l'exemple courant limité par la borne supérieure S_{couv} et la borne inférieure I_{couv}. Cet espace est constitué de toutes les couvertures de Q en utilisant $\mathcal{T}$.

par la construction d'une description E_1 de meilleur rest étant donnée une description E_2.

Ce lemme nous permet de déduire une nouvelle borne supérieure de l'espace de recherche. En effet, grâce au membre de droite de l'équivalence et à la caractérisation de la subsomption structurelle dans $\mathcal{ALN}$ (voir le théorème 4.2.1 page 49), on vérifie facilement le lemme 4.3.4 qui dit que si une description E_1 est subsumée par une description E_2, alors le rest de E_2 est subsumé par le rest de E_1. Cette propriété d'anti-monotonie est évidemment très utile dès lors qu'il s'agit de trouver des bornes à l'espace de recherche.

Lemme 4.3.4
Soit Q une $\mathcal{ALN}$-description. Soit $\mathcal{T}$ une $\mathcal{ALN}$-terminologie avec $\mathcal{S}_\mathcal{T}$ l'ensemble des concepts définis S_i. Soient E_1 et E_2 deux couvertures de Q en utilisant $\mathcal{T}$. On a :

$$E_1 \sqsubseteq E_2 \Rightarrow Rest_{E_2}(Q) \sqsubseteq Rest_{E_1}(Q)$$

La démonstration découle facilement du lemme 4.3.3.

Les éléments de S_{couv} étant maximaux par rapport à l'inclusion, les conjonctions correspondantes sont minimales par rapport à la subsomption, et donc leur rest est maximal par rapport à la subsomption. Ainsi, parmi les éléments de S_{couv} se trouvent toutes les plus grandes couvertures de Q ayant le(s) rest maximal(-aux) par rapport à la subsomption. Il suffit donc de comparer les rest des éléments de S_{couv} deux à deux (selon le lemme 4.3.3) pour obtenir les plus grandes couvertures de rest maximal. On appelle S_{rest} le sous-ensemble de S_{couv} obtenu.

$S_{rest} = \{E' \in S_{couv} \mid E'$ a un rest maximal par rapport à la subsomption$\}$

Après S_{cons} et S_{couv}, S_{rest} constitue donc une troisième borne supérieure de l'espace de recherche.

Voyons maintenant comment le lemme 4.3.3 permet de déduire une nouvelle borne inférieure de l'espace de recherche. Il implique en effet le lemme 4.3.5 qui caractérise en termes de clauses l'équivalence entre deux rest. Ainsi les rest de deux couvertures sont équivalents si et seulement si les clauses de Q couvertes par E_1 sont les mêmes que celles couvertes par E_2 et si les clauses de Q couvertes indirectement le sont par les mêmes clauses dans E_1 et E_2.

Lemme 4.3.5
Soit Q une $\mathcal{ALN}$-description. Soit $\mathcal{T}$ une $\mathcal{ALN}$-terminologie avec $\mathcal{S}_\mathcal{T}$ l'ensemble des concepts définis S_i. Soient E_1 et E_2 deux couvertures de Q en utilisant $\mathcal{T}$. On a $Rest_{E_1}(Q) \equiv Rest_{E_2}(Q)$ si et seulement si :
$\{c_Q \in \widehat{\mathcal{G}}^{\#}_Q \mid c_Q$ *est couverte par* $E_2\} = \{c_Q \in \widehat{\mathcal{G}}^{\#}_Q \mid c_Q$ *est couverte par* $E_1\}$ *et*
$\forall c_Q \in \widehat{\mathcal{G}}^{\#}_Q$, $\{c_{E_2} \in \widehat{\mathcal{G}}^{\#}_{E_2} \mid c_{E_2}$ *couvre indirectement* $c_Q\} = \{c_{E_1} \in \widehat{\mathcal{G}}^{\#}_{E_1} \mid c_{E_1}$ *couvre indirectement* $c_Q\}$

La démonstration découle facilement du lemme 4.3.3.

Grâce au lemme 4.3.5, on peut donc calculer, pour chaque élément E^* de S_{rest}, l'ensemble $R_{eq}(E^*)$ de ses plus petits sous-ensembles de rest équivalent, et ce en travaillant toujours au niveau des clauses (et non en calculant explicitement et en comparant des rest).

$R_{eq}(E^*)$= $\{X \subseteq E^* \mid X$ est minimal par rapport à l'inclusion et $Rest_{E^*}(Q) \equiv Rest_X(Q)\}$

En résumé, S_{rest} contient toutes les plus grandes couvertures de rest maximal, constituant ainsi une borne supérieure de l'espace de recherche, et l'union de tous les $R_{eq}(E^*)$ contient tous les plus petites couvertures de rest maximal, constituant ainsi une borne inférieure de l'espace de recherche. Cette borne inférieure est nommée I_{rest}. Le théorème suivant résume ceci en donnant une caractérisation des couvertures ayant un rest maximal par rapport à la subsomption.

$$I_{rest} = \bigcup_{E^* \in S_{rest}} R_{eq}(E^*)$$

Théorème 4.3.2 (Caractérisation des couvertures $\mathcal{ALN}$ de rest maximal)
Soit Q une $\mathcal{ALN}$-description. Soit $\mathcal{T}$ une $\mathcal{ALN}$-terminologie avec $\mathcal{S}_\mathcal{T}$ l'ensemble des concepts définis S_i. Soient S_{rest} et I_{rest} les ensembles définis précédemment, contenant respectivement les plus grandes et plus petites couvertures de rest maximal.

Un ensemble E_{rest} de S_i (i.e. $E_{rest} \subseteq \mathcal{S}_\mathcal{T}$) est une couverture $\mathcal{ALN}$ de Q en utilisant $\mathcal{T}$ qui maximise le rest par rapport à la subsomption si et seulement si :

- *E_{rest} est un sous-ensemble d'un élément E^* de S_{rest} et*
- *E_{rest} est un sur-ensemble d'un élément X de I_{rest}*

Exemple courant 4
En comparant deux à deux, grâce au lemme 4.3.3, les rest des éléments de S_{couv}, on peut montrer que, parmi les six éléments de S_{couv}, seuls E_2, E_4 et E_6 ont un rest maximal (le même pour les trois). Ainsi, $S_{rest} = \{E_2, E_4, E_6\}$. C'est la troisième borne supérieure de l'espace de recherche. Grâce au lemme 4.3.5, on calcule les ensembles $R_{eq}(E^*)$, avec $E^* = E_2$, $E^* = E_4$ et $E^* = E_6$:
– $R_{eq}(E_2)$= $\{\{T_3, H_1, H_2\}, \{T_2, T_3, H_1\}\}$ regroupe tous les plus petits sous-ensembles de E_2 de rest équivalent,
– $R_{eq}(E_4)$= $\{\{T_3, H_1, H_2\}\}$ et
– $R_{eq}(E_6)$= $\{\{T_1, T_2, H_1, H_2\}\}$.
Ainsi la deuxième borne inférieure de l'espace de recherche est :
I_{rest}= $\{\{T_3, H_1, H_2\}, \{T_2, T_3, H_1\}, \{\{T_1, T_2, H_1, H_2\}\}$.

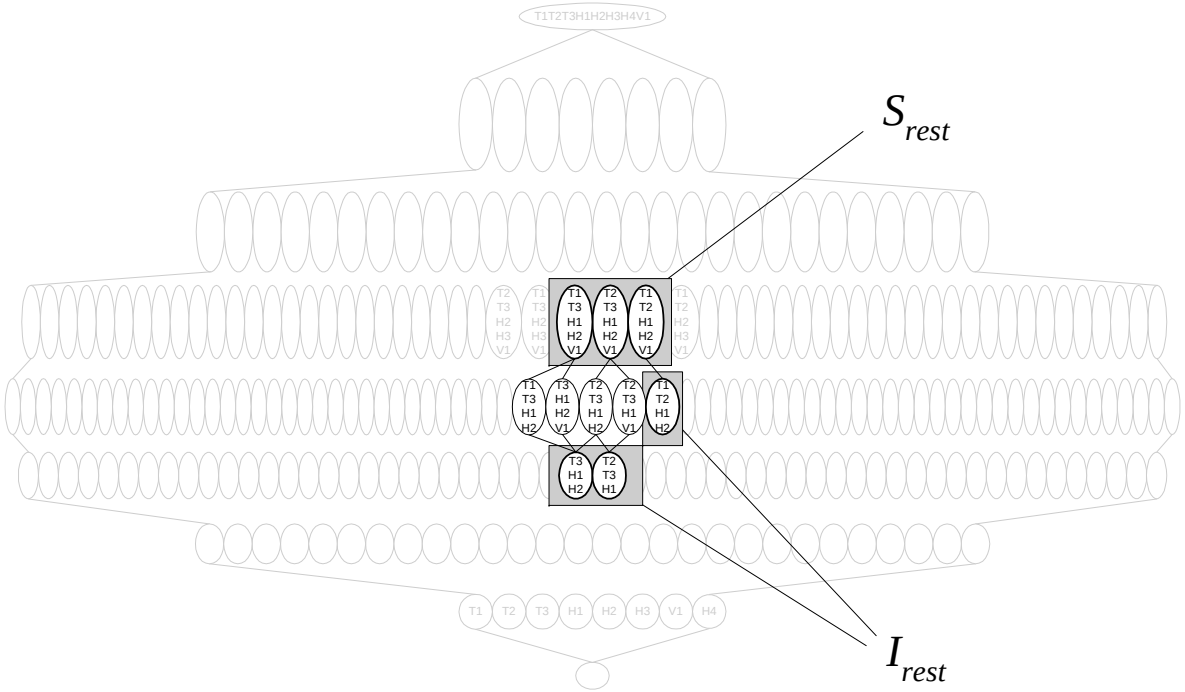


FIG. 4.4 – L'espace de recherche des meilleures couvertures de l'exemple courant limité par la borne supérieure S_{rest} et la borne inférieure I_{rest}. Cet espace est constitué de toutes les couvertures de Q en utilisant $\mathcal{T}$ ayant un rest maximal par rapport à la subsomption.

La figure 4.4 montre l'espace de recherche nouvellement borné. ◦

4.3.4 Conditions (e) et (f) de la définition 4.2.6

Terminer la résolution du problème de découverte des meilleures couvertures $\mathcal{ALN}$ comme elle a été entamée reviendrait à fournir une caractérisation en termes de clauses de la condition (e) de la définition 4.2.6, pour obtenir les ensembles de S_i ayant les miss les plus grands par rapport à la subsomption. On pourrait alors découvrir ceux-ci sans avoir à comparer les miss des ensembles de S_i restants dans l'espace de recherche, mais en comparant simplement leurs clauses.

Dans l'état actuel de nos travaux, une telle caractérisation reste à trouver. La principale difficulté dans l'obtention d'une telle caractérisation est le fait que, contrairement au rest, le miss ne présente pas de propriété de monotonie ou d'anti-monotonie. Ainsi à partir de deux ensembles de S_i, notés E_1 et E_2, tels que $E_1 \subseteq E_2$, aucune relation d'équivalence ou de subsomption ne peut être déduite entre $Miss_{E_1}(Q)$ et $Miss_{E_2}(Q)$, comme montré dans l'exemple 19 ci-dessous.

Exemple 19

Nous présentons ici trois situations différentes dans lesquelles nous ajoutons à une couverture E un S_i et comparons $Miss_E(Q)$ avec $Miss_{E \sqcap S_i}(Q)$: dans la première, nous obtenons $Miss_{E \sqcap S_i}(Q) \sqsubseteq Miss_E(Q)$; dans la seconde $Miss_{E \sqcap S_i}(Q)$ et $Miss_E(Q)$ sont incomparables ; et dans la dernière $Miss_{E \sqcap S_i}(Q) \sqsupseteq Miss_E(Q)$.

Situation 1 :
$E \equiv \forall R_1.(\geq 1\ R_2) \sqcap \forall R_1 R_2.(\leq 0\ R_3)$
$Q \equiv \forall R_1 R_2 R_3.P$
$S_i \equiv (\leq 0\ R_1)$
On a :
$Miss_E(Q) \equiv \{\forall R_1.(\geq 1\ R_2) \sqcap \forall R_1 R_2 R_3.\neg P\}$
$Miss_{E \sqcap S_i}(Q) \equiv \{\forall R_1.(\geq 1\ R_2) \sqcap \forall R_1 R_2 R_3.\neg P \sqcap \forall R_1 R_2.(\geq 1\ R_3)\}$
D'où : $Miss_{E \sqcap S_i}(Q) \sqsubseteq Miss_E(Q)$.

Situation 2 :
$E \equiv \forall R_1.(\geq 6\ R_2) \sqcap \forall R_1 R_2.(\leq 0\ R_3)$
$Q \equiv \forall R_1 R_2 R_3.P$
$S_i \equiv (\leq 0\ R_1)$
On a :
$Miss_E(Q) \equiv \{\forall R_1.(\geq 6\ R_2) \sqcap \forall R_1 R_2 R_3.\neg P\}$
$Miss_{E \sqcap S_i}(Q) \equiv \{\forall R_1.(\geq 1\ R_2) \sqcap \forall R_1 R_2 R_3.\neg P \sqcap \forall R_1 R_2.(\geq 1\ R_3)\}$
D'où : $Miss_{E \sqcap S_i}(Q)$ et $Miss_E(Q)$ sont incomparables.

Situation 3 :
$E \equiv \forall R_1.(\geq 6\ R_2)$
$Q \equiv \forall R_1.(\leq 3\ R_2)$
$S_i \equiv (\leq 0\ R_1)$
On a :
$Miss_E(Q) \equiv \{\forall R_1.(\geq 6\ R_2)\}$
$Miss_{E \sqcap S_i}(Q) \equiv \{\forall R_1.(\geq 4\ R_2)\}$
D'où : $Miss_{E \sqcap S_i}(Q) \sqsupseteq Miss_E(Q)$.

$\circ$

Ainsi, on ne peut pas déduire de nouvelle borne inférieure ou supérieure à l'espace de recherche. La seule solution qui reste est d'examiner tous les ensembles, c'est-à-dire de calculer leur miss et de comparer ces miss. Notons que, comme on connaît toutes les clauses possibles d'une conjonction normalisée de S_i, on peut calculer ces miss à partir de ces clauses. La phase de normalisation par application des règles de la section 4.2.1 peut donc être évitée. Par contre, la comparaison des miss doit se faire, cette fois, à partir de la définition 4.2.4 de la subsomption entre ensembles de descriptions, et non à partir de comparaisons entre couples de clauses.

La dernière condition définissant les meilleures couvertures est la condition (f) impliquant la minimisation des ensembles de S_i obtenus (pour ne pas avoir deux S_i équivalents dans une même couverture). Pour la vérifier, il suffit de trier les couvertures de rest et miss maximal par rapport à l'inclusion et de ne garder que les plus petites.

Exemple courant 5
Pour terminer l'exemple courant, le calcul des miss des 10 solutions possibles restant à la fin de l'étape précédente conclut que les seules ensembles de S_i de miss minimal sont $\{T_3, H_1, H_2\}$, $\{T_2, T_3, H_1\}$ et $\{T_1, T_2, H_1, H_2\}$, i.e. les éléments de I_{rest}. Ces ensembles étant aussi minimaux par rapport à l'inclusion, ce sont les meilleures couvertures $\mathcal{ALN}$ de Q en utilisant $\mathcal{T}$. On peut voir ce résultat à la figure 4.5. $\circ$

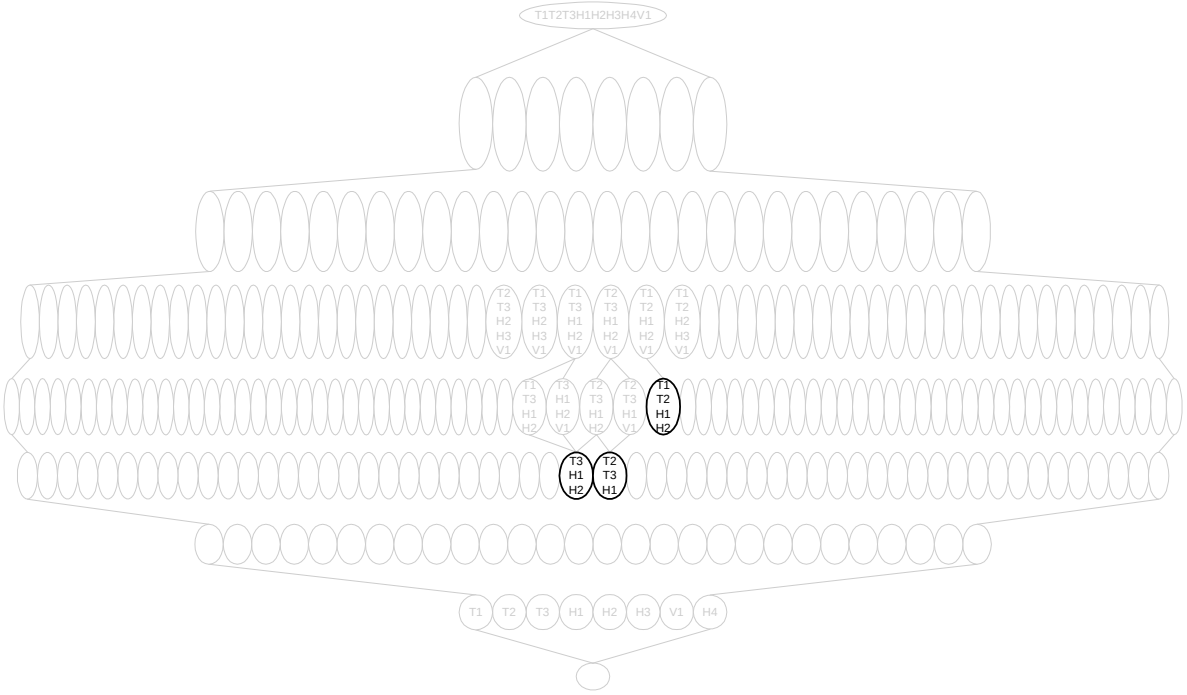


FIG. 4.5 – Les meilleures couvertures $\mathcal{ALN}$ de Q en utilisant $\mathcal{T}$ dans l'exemple courant.

4.4 Calcul des meilleures couvertures dans $\mathcal{ALN}$

Dans cette section, on présente l'algorithme *compute*$\mathcal{ALN}$*BCov* qui permet de trouver toutes les meilleures couvertures d'un concept Q étant donnée une terminologie $\mathcal{T}$ de concepts définis S_i. On rappelle que $\mathcal{S}_\mathcal{T}$ est l'ensemble des S_i de $\mathcal{T}$ et $\mathcal{S}_{\mathcal{T}_Q}$ l'ensemble $\mathcal{S}_\mathcal{T} \cup \{Q\}$.

Les principales étapes de *compute*$\mathcal{ALN}$*BCov* correspondent au calcul des bornes supérieures et inférieures précédemment définies. Ainsi, dans la section 4.4.1, nous expliquons comment obtenir la borne supérieure S_{cons} au travers du calcul de tous les cas d'inconsistance explicite. A la section 4.4.2, on détaille le calcul de tous les cas d'inconsistance implicite. Ceci permet d'obtenir C_{dir} et C_{indir} et la borne inférieure I_{couv} qui en découle, ainsi que la deuxième borne supérieure S_{couv}. Dans les sections 4.4.3 et 4.4.4, on montre comment obtenir la troisième borne supérieure S_{rest} et la deuxième borne inférieure I_{rest} constituée par l'union des $R_{eq}(E^*)$ pour tous les éléments E^* de S_{rest}. Comme dans la section 4.3, chaque calcul sera illustré par son application à l'exemple courant.

Le problème du calcul des transversaux minimaux d'un hypergraphe se pose plusieurs fois au cours des calculs de bornes. A la suite du chapitre 3, on utilise l'algorithme 2 des persistants, page 37. On verra que la structure itérative de cet algorithme, où chaque itération correspond à une arête de l'hypergraphe, permet un calcul simultané des inconsistances explicites et implicites.

4.4.1 Recherche des inconsistances explicites : calcul de S_{cons}

Comme nous l'avons vu dans la section 4.3.1, nous proposons de découvrir les inconsistances explicites entre les S_i et Q en découvrant celles entre leurs clauses. A partir des inconsistances explicites entre clauses, nous calculons :

– l'ensemble Inc de tous les plus petits ensembles de S_i de $\mathcal{S}_\mathcal{T}$ dont la conjonction avec Q est inconsistante, puis
– l'ensemble S_{cons} de tous les plus grands ensembles de S_i dont la conjonction avec Q est consistante.

Calcul de Inc

La démarche est la suivante :
– On cherche d'abord tous les plus petits ensembles de clauses (prises dans toutes les clauses des S_i et de Q) dont la conjonction est inconsistante. Pour cela, nous utilisons la caractérisation du lemme 4.2.2 page 51. Ainsi, pour chaque clause c' de chaque S_i et de Q, nous calculons les clauses $c_\uparrow$ de l'approximation faible de $\neg c'$. Puis, nous regardons dans les clauses restantes des S_i et de Q s'il existe au moins une clause c^* subsumée et de même profondeur que chaque clause $c_\uparrow$. Si c'est le cas, alors il existe une inconsistance explicite entre c' et les c^*.
– Dès lors, pour cette clause c' on peut contruire un hypergraphe $\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}}$ ayant comme sommets les S_i et Q, et comme arêtes c' et les $c_\uparrow$. Un sommet est dans une arête si le S_i (ou Q) correspondant possède c' ou une clause c^* subsumée et de même profondeur que la clause $c_\uparrow$ correspondant à l'arête. Chaque transversal minimal représente alors un plus petit ensemble de S_i (par rapport à l'inclusion) qui contient c' et les c^* et qui est donc inconsistant par conjonction.
– Le calcul de tous les transversaux minimaux de $\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}}$ pour toute clause c' des S_i et de Q assure que l'on trouve tous les cas d'inconsistance explicite et les plus petits ensembles de S_i qui les contiennent. On regroupe tous ces ensembles de S_i dans l'ensemble nommé Inc_Q. Inc_Q contient donc tous les plus petits ensembles de S_i de $\mathcal{S}_{\mathcal{T}_\mathcal{Q}}$ dont la conjonction est inconsistante.

Le théorème 4.4.1 résume ces trois étapes pour l'obtention de Inc_Q.

Théorème 4.4.1
Soit $\mathcal{S}_{\mathcal{T}_\mathcal{Q}}$ un ensemble de S_i et $\mathcal{C}_Q$ l'ensemble des clauses correspondantes (i.e. $\mathcal{C}_Q = \bigcup_{S_i \in \mathcal{S}_{\mathcal{T}_\mathcal{Q}}} \widehat{\mathcal{G}}^{\#}_{S_i}$).

Pour toute clause $c' \in \mathcal{C}_Q$, on note $\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}}$ l'hypergraphe construit ainsi :
– *l'ensemble des sommets est $\Sigma_{c'}$: chaque sommet est un S_i de $\mathcal{S}_{\mathcal{T}_\mathcal{Q}}$ (et réciproquement),*
– *l'ensemble des arêtes est $\Gamma_{c'}$: c' est une arête et les autres arêtes sont les $c_\uparrow \in \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c')}$,*
– *$S_i \in c_\uparrow \Leftrightarrow \exists c^* \in \widehat{\mathcal{G}}^{\#}_{S_i} \mid c^* \sqsubseteq c_\uparrow$ et $prof(c^*) = prof(c_\uparrow)$*

On a alors :

$$Inc_Q = \{E_\bot \subseteq \mathcal{S}_{\mathcal{T}_\mathcal{Q}} \mid \bigsqcap_{S_i \in E_\bot} S_i \equiv \bot \text{ et } \forall S_j \in E_\bot, \bigsqcap_{S_i \in E_\bot \setminus \{S_j\}} S_i \not\equiv \bot \} = Min_\subseteq \left\{ X \mid X \in \bigcup_{c' \in \mathcal{C}_Q} (Tr(\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}})) \right\}$$

La démonstration de ce théorème est donnée dans l'annexe A.7 page 157.

Inc_Q contient tous les plus petits ensembles de S_i de $\mathcal{S}_{\mathcal{T}_\mathcal{Q}}$ (incluant Q) dont la conjonction est inconsistante. Or, on veut obtenir Inc qui contient tous les plus petits ensembles de S_i de $\mathcal{S}_\mathcal{T}$ (n'incluant pas Q) dont la conjonction avec Q est inconsistante. Il reste donc à supprimer

Q des ensembles de S_i de Inc_Q et à ne garder que les ensembles de S_i minimaux par rapport à l'inclusion pour obtenir Inc.

$Inc = Min_{\subseteq}\{E_\perp \setminus \{Q\} \mid E_\perp \in Inc_Q\}$.

Exemple courant 6

Dans l'exemple courant de la section 4.3.1 page 60, nous avons vu que la conjonction $T_1 \sqcap T_2 \sqcap T_3$ contenait l'inconsistance explicite suivante :
$\forall escale.\forall mag. \geq 1 DutyFree \sqcap \forall escale.\forall mag. \leq 0 DutyFree \sqcap \forall escale. \geq 1 mag \sqcap \geq 1 escale \equiv \perp$

On découvre cette inconsistance en considérant la clause $c' = \forall escale.\forall mag. \geq 1 DutyFree$ de T_1. Les clauses de l'approximation faible de $\neg c'$ sont :

$$\begin{aligned}\widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c')} = \quad & \{c^1_\uparrow = \forall escale.\forall mag. \leq 0 DutyFree,\\ & c^2_\uparrow = \forall escale. \geq 1 mag,\\ & c^3_\uparrow = \geq 1 escale\}\end{aligned}$$

Or il se trouve que T_2 possède la clause $\forall escale.\forall mag. \leq 0 DutyFree$ (qui est donc bien subsumée par $c^1_\uparrow$ et de même profondeur, puisque c'est la même clause), T_1 possède $c^2_\uparrow$ et T_3 possède $c^3_\uparrow$. L'hypergraphe $\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}}$ qui en découle est constitué ainsi :

- sommets : $\Sigma_{c'} = \{T_1, T_2, T_3, H_1, H_2, H_3, H_4, V_1, Q\}$
- arêtes : $\Gamma_{c'} = \{c', c^1_\uparrow, c^2_\uparrow, c^3_\uparrow\}$
- l'arête c' contient le sommet T_1, $c^1_\uparrow$ contient T_2, $c^2_\uparrow$ contient T_1 et $c^3_\uparrow$ contient T_3.

Il n'y a qu'un seul transversal minimal dans cet hypergraphe qui est $\{T_1, T_2, T_3\}$. Cela signifie que $T_1 \sqcap T_2 \sqcap T_3 \equiv \perp$.

Un autre exemple de découverte d'inconsistance explicite est celui impliqué par l'examen de la clause $c' = \geq 3 etoile$ de H_4. Les clauses de l'approximation faible de $\neg c'$ sont :

$$\widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c')} = \quad \{c^1_\uparrow = \leq 2 etoile\}$$

Or il se trouve que H_1 et Q possède $c^1_\uparrow$ et H_3 possède la clause $\leq 1 etoile$ qui est subsumée par $c^1_\uparrow$ et de même profondeur. L'hypergraphe $\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}}$ qui en découle est constitué ainsi :

- sommets : $\Sigma_{c'} = \{T_1, T_2, T_3, H_1, H_2, H_3, H_4, V_1, Q\}$
- arêtes : $\Gamma_{c'} = \{c', c^1_\uparrow\}$
- l'arête c' contient le sommet H_4 et $c^1_\uparrow$ contient H_1, Q et H_3.

Les transversaux minimaux sont les ensembles $\{H_4, H_1\}$, $\{H_4, Q\}$ et $\{H_4, H_3\}$. Cela signifie que $H_4 \sqcap H_1 \equiv \perp$, $H_4 \sqcap Q \equiv \perp$ et $H_4 \sqcap H_3 \equiv \perp$.

Un raisonnement semblable à partir de la clause $c' = \geq 2 etoile$ de H_1 aboutit à la découverte du transversal minimal $\{H_1, H_3\}$. Les autres clauses n'aboutissent à aucun transversal minimal nouveau. Ainsi :
$Inc_Q = \{\{T_1, T_2, T_3\}, \{H_4, H_1\}, \{H_4, Q\}, \{H_4, H_3\}, \{H_1, H_3\}\}$
On rappelle que ce sont les plus petits ensembles de $S_i \in \mathcal{S}_{\mathcal{T}_\mathcal{Q}}$ inconsistants par conjonction. Pour obtenir Inc, on enlève Q et on ne garde que les plus petits ensembles par inclusion. D'où :
$Inc = \{\{T_1, T_2, T_3\}, \{H_4\}, \{H_1, H_3\}\}$
On rappelle que ce sont les plus petits ensembles de $S_i \in \mathcal{S}_\mathcal{T}$ inconsistants avec Q par conjonction.

○

Calcul de S_{cons}

A partir des éléments de Inc, on peut construire un hypergraphe $\mathcal{H}_{Inc}$ dont les sommets sont les S_i de $\mathcal{S}_\mathcal{T}$ (sans Q) et les arêtes les éléments de Inc (contenant les sommets correspondants à leurs S_i). Le lemme 4.4.1 montre alors que les complémentaires dans $\mathcal{S}_\mathcal{T}$ des transversaux minimaux de $\mathcal{H}_{Inc}$ sont les plus grands ensembles de S_i ne contenant aucun élément de Inc. Comme les éléments de Inc sont les plus petits ensembles de $S_i \in \mathcal{S}_\mathcal{T}$ inconsistants avec Q par conjonction, alors les éléments de S_{cons} sont les plus grands ensembles de S_i de $\mathcal{S}_\mathcal{T}$ consistants avec Q par conjonction.

Lemme 4.4.1
Soit $\mathcal{S}_\mathcal{T}$ un ensemble de S_i. Soit Inc un ensemble d'ensembles E' de S_i de $\mathcal{S}_\mathcal{T}$ (i.e. $E' \subseteq \mathcal{S}_\mathcal{T}, \forall E'$). Soit l'hypergraphe $\mathcal{H}_{Inc} = (\mathcal{S}_\mathcal{T}, Inc)$. On a :

$$S_{cons} = \{E \subseteq \mathcal{S}_\mathcal{T} \mid \big(\forall E' \in Inc, E' \not\subseteq E \text{ et } E \text{ maximal par rapport à } \subseteq\big)\}$$
$$= \{\mathcal{S}_\mathcal{T} \setminus X \mid X \in Tr(\mathcal{H}_{Inc})\}$$

La démonstration de ce lemme est donnée dans l'annexe A.8 page 159.

Exemple courant 7
On rappelle que l'on a $Inc = \{\{T_1, T_2, T_3\}, \{H_4\}, \{H_1, H_3\}\}$. Les transversaux minimaux de $\mathcal{H}_{Inc}$ sont les ensembles :
$\{\{T_1, H_1, H_4\}, \{T_1, H_3, H_4\}, \{T_2, H_1, H_4\}, \{T_2, H_3, H_4\}, \{T_3, H_1, H_4\}, \{T_3, H_3, H_4\}\}$
Les complémentaires des transversaux minimaux de $\mathcal{H}_{Inc}$ sont les ensembles constituant S_{cons} :
$S_{cons} = \{E_1 = \{T_2, T_3, H_2, H_3, V_1\}, E_2 = \{T_2, T_3, H_1, H_2, V_1\}, E_3 = \{T_1, T_3, H_2, H_3, V_1\},$
$E_4 = \{T_1, T_3, H_1, H_2, V_1\}, E_5 = \{T_1, T_2, H_2, H_3, V_1\}, E_6 = \{T_1, T_2, H_1, H_2, V_1\}\}$
On rappelle que chacun de ces ensembles est maximal par rapport à l'inclusion et consistant avec Q par conjonction (et il n'y en a pas d'autre). ∘

4.4.2 Recherche des inconsistances implicites : calcul de C_{dir}, C_{indir}, I_{couv} et S_{couv}

Comme nous l'avons vu dans la section 4.3.2, nous proposons de découvrir les inconsistances implicites dans les conjonctions de S_i afin de déterminer tous les cas où une clause de Q est couverte par un ensemble de S_i (sans perte de généralité, on se limite aux plus petits ensembles de S_i). Cela permet de construire les ensembles C_{dir} et C_{indir} dont on tire la première borne inférieure I_{couv}, ainsi que l'ensemble S_{couv} qui constitue la deuxième borne supérieure. On rappelle que C_{dir} contient des triplets (c, H, c_Q) tels que H est un plus petit ensemble de S_i contenant c et c couvre directement c_Q qui est une clause de Q, que C_{indir} contient des triplets (c, H, c_Q) tels que H est un plus petit ensemble de S_i contenant c et c couvre indirectement c_Q, que I_{couv} contient les plus petites couvertures de Q et que S_{couv} contient les plus grandes couvertures de Q.

Si on se réfère à la caractérisation des inconsistances implicites, donnée au lemme 4.2.2 page 51, on constate qu'une inconsistance implicite est une sorte d'inconsistance explicite à laquelle il manque au moins une clause. Autrement dit, quand on enlève au moins une clause d'un ensemble minimal de clauses, contenant au moins 3 clauses, dont la conjonction est le concept $\bot$, on peut obtenir un cas d'inconsistance implicite et donc une clause d'exclusion.

Dès lors, on constate que le calcul des inconsistances explicites détaillé dans la section 4.4.1 peut être réutilisé pour trouver les inconsistances implicites. On rappelle que ce calcul est composé de plusieurs calculs de transversaux minimaux pour les hypergraphes construits à partir

de chaque clause c' des S_i et de Q. Pour chaque clause c', on va pouvoir utiliser ce calcul de transversaux minimaux pour trouver les cas d'inconsistances implicites impliquant c'.

Soit une clause c' d'un S_i ou de Q. Le premier cas d'inconsistance implicite impliquant c' est celui où une arête de l'hypergraphe $\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}}$ est vide. Cela traduit le fait qu'il manque au moins une clause pour obtenir une inconsistance explicite.

Exemple courant 8

On recherche les inconsistances explicites impliquant la clause $c' = \forall tv.\forall chaine.Satellite$ de H_1. Reprenons la méthode de découverte des inconsistances explicites vue précédemment. Les clauses de l'approximation faible de $\neg c'$ sont :

$\widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c')} = \{c^1_\uparrow = \forall tv.\forall chaine.\neg Satellite,$
$c^2_\uparrow = \forall tv. \geq 1 chaine,$
$c^3_\uparrow = \geq 1 tv\}$

Il se trouve que H_2 possède $c^1_\uparrow$, H_1 possède la clause $\forall tv. \geq 10 chaine$ qui est subsumée par $c^2_\uparrow$ et de même profondeur, mais par contre aucun S_i ne possède de clause subsumée par et de même profondeur que $c^3_\uparrow$. L'hypergraphe $\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}}$ qui en découle est constitué ainsi :

– sommets : $\Sigma_{c'} = \{T_1, T_2, T_3, H_1, H_2, H_3, H_4, V_1, Q\}$
– arêtes : $\Gamma_{c'} = \{c', c^1_\uparrow, c^2_\uparrow, c^3_\uparrow\}$
– l'arête c' contient le sommet H_1, $c^1_\uparrow$ contient H_2, $c^2_\uparrow$ contient H_1, mais $c^3_\uparrow$ est vide.

Il n'y a donc pas de transversaux minimaux à $\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}}$. Par contre si on considère cet hypergraphe sans la dernière arête (i.e. une version incomplète de $\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}}$), alors on a un transversal minimal qui est l'ensemble $\{H_1, H_2\}$. Ceci signifie que $H_1 \sqcap H_2$ contient une inconsistance implicite en la conjonction $\left(c' \sqcap c^1_\uparrow \sqcap \forall tv. \geq 10 chaine\right)$ dont le résultat est la clause d'exclusion $\leq 0tv$. On voit bien qu'il manque ici une clause subsumée par et de même profondeur que $\geq 1tv$ pour obtenir $\bot$ par conjonction. ∘

Le second cas d'inconsistance implicite impliquant c' généralise le premier. Supposons qu'on examine un calcul de transversaux minimaux qui aboutit (i.e. on en déduit au moins une inconsistance explicite). On a dit précédemment qu'on utilisait l'algorithme 2, page 37, pour le résoudre. Cet algorithme est itératif et, à chaque itération, ajoute une nouvelle arête à l'hypergraphe $\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}}$ et calcule les transversaux minimaux de cet hypergraphe incomplet (toutes les arêtes n'ayant pas encore été ajoutées). Supposons qu'un ensemble X de S_i soit un transversal minimal à la fin de l'itération k, et ne soit plus un transversal minimal à la fin de l'itération $k+1$. Cela signifie que les S_i de X ne contiennent aucune clause subsumée par et de même profondeur que la clause correspondant à l'arête $k+1$, car sinon X serait un transversal minimal de l'itération $k+1$. Si on a ajouté les arêtes dans l'ordre des clauses de l'approximation faible de $\neg c'$[17], et si on a examiné au moins trois arêtes (i.e. $k \geq 2$ puisqu'il faut au moins deux S_i différents pour avoir une inconsistance implicite), alors cela signifie aussi que la conjonction des S_i de X contient une inconsistance implicite. Illustrons une telle situation sur l'exemple courant.

Exemple courant 9

Reprenons le cas de la clause $c' = \forall escale.\forall mag. \geq 1 DutyFree$ de T_1. Les clauses de l'approximation faible de $\neg c'$ sont, dans l'ordre voulu :

$\widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c')} = \{c^1_\uparrow = \forall escale.\forall mag. \leq 0 DutyFree,$
$c^2_\uparrow = \forall escale. \geq 1 mag,$
$c^3_\uparrow = \geq 1 escale\}$

[17] Cet ordre est donné dans la section 1.2.2 et c'est celui qui a été utilisé jusqu'ici dans l'exemple courant.

Détaillons la construction de $\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}}$ et la recherche de ses transversaux minimaux par l'algorithme 2. $\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}}$ est construit ainsi :
- sommets : $\Sigma_{c'} = \{T_1, T_2, T_3, H_1, H_2, H_3, H_4, V_1, Q\}$
- arêtes : $\Gamma_{c'} = \{c', c^1_\uparrow, c^2_\uparrow, c^3_\uparrow\}$
- les arêtes sont remplies ainsi :
  - l'arête c' contient le sommet T_1 car T_1 contient la clause c',
  - $c^1_\uparrow$ contient T_2 car T_2 possède la clause $\forall escale.\forall mag. \leq 0 DutyFree = c^1_\uparrow$,
  - $c^2_\uparrow$ contient T_1 car T_1 possède $c^2_\uparrow$ et
  - $c^3_\uparrow$ contient T_3 car T_3 possède $c^3_\uparrow$.

Voici les itérations de l'agorithme 2 pour rechercher les transversaux minimaux :

Initialisation : L'ensemble courant des transversaux minimaux est initialisé à l'ensemble vide.

Itération 1 : La première arête examinée est c'. Elle ne possède que le sommet T_1. Donc l'unique transversal minimal à la fin de cette itération est $\{T_1\}$.

Itération 2 : La deuxième arête est $c^1_\uparrow$. Elle ne possède que le sommet T_2. Le transversal minimal $\{T_1\}$ de l'itération précédente est donc un non-persistant (son intersection avec l'arête courante est vide). Donc l'unique transversal minimal à la fin de cette itération est $\{T_1, T_2\}$.

Itération 3 : La troisième arête est $c^2_\uparrow$. Elle ne possède que le sommet T_1. Le transversal minimal $\{T_1, T_2\}$ de l'itération précédente est donc un persistant : il est encore transversal minimal à la fin de cette itération (et c'est le seul). A la fin de cette itération, comme on a déjà examiné trois arêtes correspondant à trois clauses, des cas d'inconsistance implicite peuvent survenir. Si parmi les transversaux minimaux de l'itération précédente, certains ne le sont plus à la fin de cette itération (ce sont donc des non-persistants), alors la conjonction de S_i correspondante contient une inconsistance implicite. Pour être plus synthétique, on peut dire qu'à partir de cette itération les non-persistants correspondent à des cas d'inconsistance implicite. Ici il n'y a aucun non-persistant.

Itération 4 : La quatrième (et dernière) arête est $c^3_\uparrow$. Elle ne possède que T_3. Le transversal minimal de l'itération 3, qui est $\{T_1, T_2\}$, est donc un non-persistant et donc l'unique transversal minimal de cette itération est $\{T_1, T_2, T_3\}$. Comme il ne reste plus d'arête à examiner, on a $Tr(\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}}) = \{\{T_1, T_2, T_3\}\}$.

La conclusion est donc double :
- Il y a un seul non-persistant (à partir de l'itération 3) qui est $\{T_1, T_2\}$. Donc, une seule inconsistance implicite peut être déduite de c'. Elle est présente dans $T_1 \sqcap T_2$ et elle génère la clause d'exclusion $c' \sqcap c^1_\uparrow \sqcap c^2_\uparrow \equiv \leq 0 escale$.
- $\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}}$ a un seul transversal minimal. Donc, une seule inconsistance explicite peut être déduite de c'. C'est $T_1 \sqcap T_2 \sqcap T_3 \equiv \bot$.

$\circ$

En résumé, pour trouver les inconsistances implicites dans ce deuxième cas, il faut conserver les non-persistants (à partir de la troisième itération) pendant le calcul des inconsistances explicites. Ce deuxième cas généralise le premier car les arêtes vides génèrent obligatoirement des non-persistants (puisque l'intersection avec une arête vide est toujours l'ensemble vide). Ainsi, sans surcoût, on calcule avec un même algorithme les plus petits ensembles de S_i contenant toutes les inconsistances explicites et implicites. Le seul surcoût est celui du calcul des clauses d'exclusion au moment où l'on découvre les inconsistances implicites. Ce calcul est simple puisqu'on peut déduire facilement la clause d'exclusion à partir de la clause définissant la dernière arête ajoutée. Notons que l'utilisation de l'algorithme 2 telle qu'elle est présentée ici implique de garder dans les hypergraphes manipulés les arêtes vides et de ne pas traiter à part les arêtes

de cardinalité 1.

Nous ne donnons pas ici de théorème récapitulatif car une formalisation de la méthode présentée précédemment serait inutilement lourde et difficile à lire. De plus, la démonstration de la justesse et de la complétude dérive directement de la caractérisation des inconsistances implicites donnée au lemme 4.2.2. Notons seulement que la minimalité des ensembles de S_i contenant une inconsistance implicite est assurée par le fait qu'ils correspondent à des transversaux minimaux, et la complétude par le fait que l'on effectue ce calcul pour toutes les clauses c' des S_i et de Q.

A ce stade, on connaît tous les plus petits ensembles de S_i de $\mathcal{S}_{\mathcal{T}_Q}$ contenant une inconsistance implicite. Avant de pouvoir construire C_{dir} et C_{indir}, il reste à retirer les cas où Q est contenu dans un de ces ensembles de S_i. De plus, au sein d'une même conjonction E de S_i, il peut y avoir plusieurs inconsistances implicites aboutissant à plusieurs clauses d'exclusion elles-mêmes liées par des liens de subsomption. Après le calcul de ces clauses d'exclusion, il faut donc ne garder que celles qui sont minimales par rapport à la subsomption, pour une même conjonction de S_i.

En regroupant les clauses des S_i et les clauses d'exclusion précédemment calculées, on connaît toutes les clauses possibles qui peuvent appartenir à la forme normalisée d'une conjonction E de S_i quelconque. On est donc en mesure de construire les ensembles C_{dir} et C_{indir}, en testant la couverture directe ou indirecte de chaque clause de Q avec toutes les clauses des S_i et les clauses d'exclusion. Puis, à partir de C_{dir} et C_{indir}, on obtient I_{couv} en ne gardant des H de C_{dir} et C_{indir} que ceux qui sont inclus dans un élément de S_{cons}. Ensuite, l'obtention de S_{couv} est simple puisque S_{couv} regroupe les éléments de S_{cons} qui sont sur-ensembles d'au moins un ensemble H de I_{couv}. Il suffit donc de tester l'inclusion de chaque élément de S_{cons} avec chaque H de I_{couv}.

4.4.3 Maximisation des rest : calcul de S_{rest}

Nous étudions dans cette section comment calculer les plus grandes conjonctions de S_i de rest maximal constituant la troisième borne supérieure nommée S_{rest}.

S_{rest} est constitué des ensembles de S_i éléments de S_{couv} qui ont un rest maximal par rapport à la subsomption. Pour obtenir ces ensembles, nous utilisons la caractérisation en termes de clauses de la subsomption entre deux rest, donnée au lemme 4.3.3 page 65. Pour tester cette caractérisation entre chaque couple d'ensembles de S_i de S_{couv}, on a besoin de connaître les clauses des conjonctions normalisées de ces ensembles de S_i, ce qui est facilité par le fait que l'on connaît toutes les clauses possibles d'une conjonction normalisée de S_i.

Soit E' une conjonction de S_i de S_{couv}. Pour trouver toutes les clauses de sa forme normalisée (i.e. $\widehat{\mathcal{G}}^{\#}_{E'}$), on commence par regrouper toutes les clauses de chacun de ses S_i normalisés (i.e. tous les $\widehat{\mathcal{G}}^{\#}_{S_i}$, pour tout $S_i \in E'$). On ajoute les clauses d'exclusion obtenues précédemment lors du calcul des inconsistances implicites et apparaissant dans des ensembles de S_i inclus dans E'. On ne garde de toutes ces clauses que les plus petites par rapport à la subsomption (les autres disparaissant au cours de la normalisation). Les clauses restantes constituent $\widehat{\mathcal{G}}^{\#}_{E'}$.

Pour comparer les rest des E' deux à deux par rapport à la subsomption, on examine comment chaque clause de Q est couverte par les clauses de chaque E'. En comparant ces couvertures selon le lemme 4.3.3, on détermine quels E' ont les rest maximaux par rapport à la subsomption.

Exemple courant 10

Le tableau 4.5 résume l'ensemble des clauses des S_i et d'exclusion associées aux plus petits ensembles de S_i qui les contiennent. Les seules clauses d'exclusion sont c_{12} et c_{30}.
Les relations de couvertures entre clauses sont les suivantes :

Clause	T_1	T_2	T_3	H_1	H_2	H_3	H_4	Q	T_1T_2	H_1H_2	V_1
$c_1 = Vol$	X	X	X					X			
$c_2 = \forall lieuArr.Ville$	X	X	X					X			
$c_3 = \forall lieuArr.\forall situe.France$			X								
$c_4 = \forall lieuArr. \leq 100000hab$								X			
$c_5 = \forall escale. \geq 1mag$	X										
$c_6 = \forall escale.\forall mag. \geq 1DutyFree$	X										
$c_7 = \forall escale.\forall mag.BoutiqueLuxe$		X									
$c_8 = \forall escale.\forall mag. \leq 0DutyFree$		X									
$c_9 = \forall escale.\forall duree. \leq 1heure$			X					X			
$c_{10} = \forall escale.\forall duree. \leq 2heure$	X										
$c_{11} = \geq 1escale$			X								
$c_{12} = \leq 0escale$									X		
$c_{13} = \forall aniAccept.\forall transport.Cage$	X			X							
$c_{14} = \forall aniAccept.Chien$		X			X						
$c_{15} = \forall aniAccept. \leq 10kg$			X		X						
$c_{16} = \leq 0aniAccept$								X			
$c_{17} = Heberg$				X	X	X	X	X			
$c_{18} = \leq 3etoile$							X				
$c_{19} = \geq 3etoile$							X				
$c_{20} = \leq 2etoile$				X				X			
$c_{21} = \geq 2etoile$				X							
$c_{22} = \leq 1etoile$						X					
$c_{23} = \geq 1etoile$					X						
$c_{24} = \forall equip.Piscine$				X			X	X			
$c_{25} = \forall equip.Tennis$				X							
$c_{26} = \forall tv.\forall chaine.Satellite$				X							
$c_{27} = \forall tv.\forall chaine.\neg Satellite$					X						
$c_{28} = \forall tv. \geq 10chaine$				X							
$c_{29} = \forall tv.EcranPlat$		X			X			X			
$c_{30} = \leq 0tv$						X				X	
$c_{31} = Voiture$											X
$c_{32} = \forall categorie.Berline$											X
$c_{33} = \forall motorisation.4x4$											X

TAB. 4.5 – Ensemble des clauses des S_i et des clauses d'exclusion de l'exemple courant et les plus petits ensembles de S_i qui les contiennent.

- Couvertures directes :
  - $c_9 \sqsubseteq c_{10}$,
  - $c_{16} \sqsubseteq c_{13}$ et $c_{16} \sqsubseteq c_{14}$ et $c_{16} \sqsubseteq c_{15}$,
  - $c_{22} \sqsubseteq c_{20} \sqsubseteq c_{18}$,
  - $c_{19} \sqsubseteq c_{21} \sqsubseteq c_{23}$,
  - $c_{30} \sqsubseteq c_{26}$ et $c_{30} \sqsubseteq c_{27}$ et $c_{30} \sqsubseteq c_{28}$ et $c_{30} \sqsubseteq c_{29}$,
  - $c_{12} \sqsubseteq c_5$ et $c_{12} \sqsubseteq c_6$ et $c_{12} \sqsubseteq c_7$ et $c_{12} \sqsubseteq c_8$ et $c_{12} \sqsubseteq c_9$ et $c_{12} \sqsubseteq c_{10}$.
- Couvertures indirectes :
  - c_{16} est couverte indirectement par c_{13}, c_{14} et c_{15},
  - c_{30} est couverte indirectement par c_{26}, c_{27}, c_{28} et c_{29} et
  - c_{12} est couverte indirectement par c_5, c_6, c_7, c_8, c_9 et c_{10}.

A partir des relations de couverture directe (i.e. de subsomption) entre clauses, on trouve les clauses des ensembles E' de S_{couv}[18]. Pour obtenir les ensembles de clauses $\widehat{\mathcal{G}}^{\#}_{E_1}$ à $\widehat{\mathcal{G}}^{\#}_{E_6}$, on rassemble toutes les clauses de chaque S_i de E_1 à E_6 listées dans le tableau 4.5. Puis, pour avoir un ensemble de clauses normalisé, on enlève les clauses qui ne sont pas minimales par rapport à la subsomption. Le tableau 4.6 décrit les ensembles de clauses obtenus pour chaque E_1 à E_6.
A partir du tableau 4.6, on déduit le tableau 4.7 qui décrit comment chaque clause de Q est couverte par les ensembles E_1 à E_6, éléments de S_{couv}. On voit que E_2, E_4 et E_6 couvrent (directement) une clause de plus de Q que E_1, E_3 et E_5 (c'est la clause c_{24}). Donc le rest de E_2, E_4 et E_6 est plus grand que celui de E_1, E_3 et E_5. Pour départager E_2, E_4 et E_6, on regarde quelles clauses de Q ces E_i couvrent indirectement et comment. La seule clause de Q couverte indirectement est c_{16}, et elle l'est de la même façon pour E_2, E_4 et E_6 (i.e. dans les trois cas par les clauses c_{13}, c_{14} et c_{15}). Donc, d'après le lemme 4.3.3 page 65, E_2, E_4 et E_6 ont un rest equivalent. On a donc : $S_{rest} = \{E_2, E_4, E_6\}$. ◦

4.4.4 Maximisation des rest : calcul de $R_{eq}(E^*)$ et de I_{rest}

Nous étudions dans cette section comment calculer les plus petites conjonctions de S_i de rest maximal constituant la deuxième borne inférieure nommée I_{rest}. Pour chaque E^* de S_{rest}, on va calculer l'ensemble $R_{eq}(E^*)$ qui contient les plus petits sous-ensembles de E^* de rest équivalent (et donc maximal). L'union de ces $R_{eq}(E^*)$ constitue I_{rest}.

Chaque élément E^* de S_{rest} est une plus grande couverture de Q en utilisant $\mathcal{T}$ de rest maximal. Pour chacun, on cherche maintenant à trouver tous ses plus petits sous-ensembles X de rest équivalent, regroupés dans l'ensemble $R_{eq}(E^*)$. Pour ce faire, on utilise le lemme 4.3.5 qui dit que deux ensembles E_1 et E_2 de S_i ont des rest équivalents si et seulement si (i) les clauses de Q couvertes par E_1 et E_2 sont les mêmes, et (ii) les clauses de Q couvertes indirectement par E_1 et E_2 sont les mêmes et le sont par les mêmes clauses dans E_1 et E_2. Le problème est donc le calcul de ces plus petits (par inclusion) sous-ensembles X de E^* vérifiant (i) et (ii).

La démarche proposée est la suivante, pour chaque E^* élément de S_{rest} :

Etape 1 : Pour chaque clause c_Q de Q, on cherche tous les plus petits sous-ensembles Y de E^* qui couvrent c_Q de la même façon que E^* (i.e. soit directement, soit indirectement avec les mêmes clauses). On nomme $C_{egal}(E^*)$ l'ensemble de tous les couples (Y, c_Q).

$C_{egal}(E^*)$= $\{(Y, c_Q) \mid c_Q \in \widehat{\mathcal{G}}^{\#}_Q$
c_Q est couverte par E^* et Y
$Y \subseteq E^*$ et Y minimal par rapport à l'inclusion et
$\{c \in \widehat{\mathcal{G}}^{\#}_{E^*} \mid c_Q$ est couverte indir. par $c\} = \{c \in \widehat{\mathcal{G}}^{\#}_Y \mid c_Q$ est couverte indir. par $c\}\}$

[18] Dans cet exemple, les $E' \in S_{couv}$ sont les ensembles E_1 à E_6 qui sont initialement les éléments $E \in S_{cons}$.

Clause	$\widehat{\mathcal{G}}^{\#}_{Q}$	$\widehat{\mathcal{G}}^{\#}_{E_1}$	$\widehat{\mathcal{G}}^{\#}_{E_2}$	$\widehat{\mathcal{G}}^{\#}_{E_3}$	$\widehat{\mathcal{G}}^{\#}_{E_4}$	$\widehat{\mathcal{G}}^{\#}_{E_5}$	$\widehat{\mathcal{G}}^{\#}_{E_6}$
$c_1 = Vol$	X	X	X	X	X	X	X
$c_2 = \forall lieuArr.Ville$	X	X	X	X	X	X	X
$c_3 = \forall lieuArr.\forall situe.France$		X	X	X	X		
$c_4 = \forall lieuArr. \leq 100000hab$	X						
$c_5 = \forall escale. \geq 1mag$				X	X		
$c_6 = \forall escale.\forall mag. \geq 1DutyFree$				X	X		
$c_7 = \forall escale.\forall mag.BoutiqueLuxe$		X	X				
$c_8 = \forall escale.\forall mag. \leq 0DutyFree$		X	X				
$c_9 = \forall escale.\forall duree. \leq 1heure$	X	X	X	X	X		
$c_{10} = \forall escale.\forall duree. \leq 2heure$							
$c_{11} = \geq 1escale$		X	X	X	X		
$c_{12} = \leq 0escale$						X	X
$c_{13} = \forall aniAccept.\forall transport.Cage$			X	X	X	X	X
$c_{14} = \forall aniAccept.Chien$		X	X	X	X	X	X
$c_{15} = \forall aniAccept. \leq 10kg$		X	X	X	X	X	X
$c_{16} = \leq 0aniAccept$	X						
$c_{17} = Heberg$	X	X	X	X	X	X	X
$c_{18} = \leq 3etoile$							
$c_{19} = \geq 3etoile$							
$c_{20} = \leq 2etoile$	X		X		X		X
$c_{21} = \geq 2etoile$			X		X		X
$c_{22} = \leq 1etoile$		X		X		X	
$c_{23} = \geq 1etoile$		X		X		X	
$c_{24} = \forall equip.Piscine$	X		X		X		X
$c_{25} = \forall equip.Tennis$			X		X		X
$c_{26} = \forall tv.\forall chaine.Satellite$							
$c_{27} = \forall tv.\forall chaine.\neg Satellite$							
$c_{28} = \forall tv. \geq 10chaine$							
$c_{29} = \forall tv.EcranPlat$	X						
$c_{30} = \leq 0tv$		X	X	X	X	X	X
$c_{31} = Voiture$		X	X	X	X	X	X
$c_{32} = \forall categorie.Berline$		X	X	X	X	X	X
$c_{33} = \forall motorisation.4x4$		X	X	X	X	X	X

TAB. 4.6 – Clauses des E_i de S_{couv} après normalisation dans l'exemple courant.

Clause de Q	E_1	E_2	E_3	E_4	E_5	E_6
$c_1 = Vol$	CD	CD	CD	CD	CD	CD
$c_2 = \forall lieuArr.Ville$	CD	CD	CD	CD	CD	CD
$c_4 = \forall lieuArr. \leq 100000hab$						
$c_9 = \forall escale.\forall duree. \leq 1heure$	CD	CD	CD	CD	CD par c_{12}	CD par c_{12}
$c_{16} = \leq 0aniAccept$	CI par c_{14} et c_{15}	CI par c_{13}, c_{14} et c_{15}	CI par c_{13}, c_{14} et c_{15}	CI par c_{13}, c_{14} et c_{15}	CI par c_{13}, c_{14} et c_{15}	CI par c_{13}, c_{14} et c_{15}
$c_{17} = Heberg$	CD	CD	CD	CD	CD	CD
$c_{20} = \leq 2etoile$	CD par c_{22}	CD	CD par c_{22}	CD	CD par c_{22}	CD
$c_{24} = \forall equip.Piscine$		CD		CD		CD
$c_{29} = \forall tv.EcranPlat$	CD par c_{30}	CD par c_{30}	CD par c_{30}	CD par c_{30}	CD par c_{30}	CD par c_{30}

Légende :
CD signifie "couverte directement" (si aucune clause n'est précisé c'est que la clause est couverte par elle-même).
CI signifie "couverte indirectement".

TAB. 4.7 – Couvertures directes et indirectes des clauses de Q par les éléments de S_{couv}.

Etape 2 : En combinant tous les Y associés à une même clause c_Q dans les couples de $C_{egal}(E^*)$, et en ne gardant que les combinaisons de Y minimales par rapport à l'inclusion, on obtient les $R_{eq}(E^*)$ recherchés. La borne I_{rest} est ensuite obtenue comme l'union des $R_{eq}(E^*)$ pour tous les E^* de S_{rest}.

Etape 1 : construction de $C_{egal}(E^*)$

On réutilise ici les ensembles C_{dir} et C_{indir}. Soit c_Q une clause de Q couverte directement par E^*. Pour construire $C_{egal}(E^*)$, on cherche dans un premier temps les plus petits sous-ensembles Y de E^* qui couvrent directement c_Q. Pour cela, on regroupe dans l'ensemble $C_{dir}(E^*)$ tous les triplets (c, H, c_Q) de C_{dir} tels que $H \subseteq E^*$. On a donc :
$C_{dir}(E^*) = \{(c, H, c_Q) \in C_{dir} \mid H \subseteq E^*\}$
Dès lors, il suffit d'extraire des triplets (c, H, c_Q) de $C_{dir}(E^*)$ tous les couples (Y, c_Q) avec $Y = H$ et on a tous les plus petits sous-ensembles Y de E^* qui couvrent directement c_Q. Ces Y constituent les premiers éléments de $C_{egal}(E^*)$.

Soit c_Q une clause de Q couverte indirectement par E^*. Pour terminer la construction de $C_{egal}(E^*)$, on cherche dans un deuxième temps les plus petits sous-ensembles Y de E^* qui couvrent indirectement c_Q avec les mêmes clauses que E^*. Clairement, il faut connaître au prélable par quelles clauses E^* couvre indirectement c_Q. Pour cela, on regroupe dans l'ensemble $C_{indir}(E^*)$ tous les triplets (c, H, c_Q) de C_{indir} tels que $H \subseteq E^*$. On ne conserve des triplets rassemblés que ceux ayant une clause c minimale par rapport à la subsomption (puisque les autres clauses disparaissent au cours de la normalisation de E^*). On a donc :
$C_{indir}(E^*) = \{(c, H, c_Q) \in C_{indir} \mid H \subseteq E^* \text{ et } c \in \widehat{\mathcal{G}}^{\#}_{E^*}\}$
C'est à partir des triplets de $C_{indir}(E^*)$ que l'on construit le reste de l'ensemble $C_{egal}(E^*)$. Il reste à ajouter à $C_{egal}(E^*)$ les Y qui couvrent c_Q indirectement et avec les mêmes clauses que

E^*. Comme on veut que ces Y couvrent indirectement les c_Q *avec les mêmes clauses* que E^*, ils sont plus difficiles à obtenir que les premiers Y obtenus à partir de $C_{dir}(E^*)$. Pour chaque couple (c, c_Q) que l'on trouve dans au moins un triplet de $C_{indir}(E^*)$, on regroupe les H des triplets de $C_{indir}(E^*)$ contenant ce couple (c, c_Q). On fait ensuite le produit cartésien de tous les ensembles de H. On ontient des ensembles d'ensembles de S_i (puisque chaque H est un ensemble de S_i). En fusionnant tous les S_i pour chaque ensemble d'ensembles de S_i, et en ne gardant des ensembles fusionnés que les plus petits par inclusion, on trouve les Y recherchés. On complète alors $C_{egal}(E^*)$ avec les couples (Y, c_Q).

Le lemme 4.4.2 récapitule cette méthode de construction de l'ensemble $C_{egal}(E^*)$.

Lemme 4.4.2
Soit $\mathcal{T}$ une $\mathcal{ALN}$-terminologie et Q une $\mathcal{ALN}$-description de concept. Soient C_{dir}, C_{indir} et S_{rest} les ensembles précédemment definis. Soit E^* un élément de S_{rest}, et $C_{dir}(E^*)$ et $C_{indir}(E^*)$ les ensembles définis et construits comme précédemment.

L'ensemble $C_{egal}(E^*)$ défini comme suit :

$C_{egal}(E^*)= \{(Y, c_Q) \mid c_Q \in \widehat{\mathcal{G}}^{\#}_Q$
c_Q est couverte par E^* et Y
$Y \subseteq E^*$ et Y minimal par rapport à l'inclusion et
$\{c \in \widehat{\mathcal{G}}^{\#}_{E^*} \mid c_Q$ est couverte indir. par $c\} = \{c \in \widehat{\mathcal{G}}^{\#}_Y \mid c_Q$ est couverte indir. par $c\}$
$\}$

peut être construit de la manière suivante :

$C_{egal}(E^*)= \{(Y, c_Q) \mid \exists (c, H, c_Q) \in C_{dir}(E^*)$ *et* $H = Y\}$
$\cup \{(Y, c_Q) \mid \exists (c', H', c_Q) \in C_{indir}(E^*)$ *et* $Y \in Min_{\subseteq}(\times_{\widetilde{c} \in \widehat{\mathcal{G}}^{\#}_{E^*}} \{\widetilde{H} | (\widetilde{c}, \widetilde{H}, c_Q) \in C_{indir}(E^*)\})\}$

Dans l'expression précédente, après le produit cartésien, on sous-entend que l'on fusionne en un seul ensemble de S_i les $\widetilde{H}$ des tuples du produit cartésien.

La démonstration est donnée dans l'annexe A.9.

Exemple courant 11
Nous prenons l'exemple de $E_2 = \{T_2, T_3, H_1, H_2, V_1\}$. Les ensembles $C_{dir}(E_2)$, $C_{indir}(E_2)$ et $C_{egal}(E_2)$ sont donnés au tableau 4.8. On rappelle que $C_{dir}(E_2)$ (respectivement $C_{indir}(E_2)$) est le sous-ensemble de C_{dir} (respectivement C_{indir}) ne contenant que les triplets (c, H, c_Q) avec $H \subseteq E_2$ (respectivement avec $H \subseteq E_2$ et $c \in \widehat{\mathcal{G}}^{\#}_{E_2}$).
Nous illustrons maintenant comment $C_{egal}(E_2)$ est construit. D'abord, tous les couples (H, c_Q) des triplets (c, H, c_Q) de $C_{dir}(E_2)$ deviennent des couiples (Y, c_Q) éléments de $C_{egal}(E_2)$. Les autres couples (Y, c_Q) sont obtenus de la manière suivante. Dans les triplets (c, H, c_Q) de $C_{indir}(E_2)$, il y a trois couples différents (c, c_Q) qui sont
– $(\forall aniAccept.\forall transport.Cage, \leq 0 aniAccept)$ lié à H_1,
– $(\forall aniAccept.Chien, \leq 0 aniAccept)$ lié à H_2 et T_2, et
– $(\forall aniAccept. \leq 10kg, \leq 0 aniAccept)$ lié à H_2 et T_3.
Pour chaque couple, on regroupe les H correspondants (ici respectivement H_1, H_2 et T_2, et H_2 et T_3) et on fait le produit cartésien de ces ensembles de H. Ainsi on fait le produit cartésien suivant :
$\{\{H_1\}\} \times \{\{T_2\}, \{H_2\}\} \times \{\{T_3\}, \{H_2\}\}$. On obtient les tuples suivants :
– $\{\{H_1\}, \{H_2\}, \{H_2\}\}$,
– $\{\{T_2\}, \{H_1\}, \{H_2\}\}$,
– $\{\{T_3\}, \{H_1\}, \{H_2\}\}$ et
– $\{\{T_2\}, \{T_3\}, \{H_1\}\}$.

$C_{dir}(E_2) =$	$\{(Vol, T_2, Vol),$ $(Vol, T_3, Vol),$ $(\forall lieuArr.Ville, T_2, \forall lieuArr.Ville),$ $(\forall lieuArr.Ville, T_3, \forall lieuArr.Ville),$ $(Heberg, H_1, Heberg),$ $(Heberg, H_2, Heberg),$ $(\leq 2etoile, H_1, \leq 2etoile),$ $(\forall equip.Piscine, H_1, \forall equip.Piscine),$ $(\forall tv.EcranPlat, T_2, \forall tv.EcranPlat),$ $(\forall tv.EcranPlat, H_2, \forall tv.EcranPlat),$ $(\forall escale.\forall duree. \leq 1heure, T_3, \forall escale.\forall duree. \leq 1heure),$ $(\leq 0tv, H_1H_2, \forall tv.EcranPlat)\}$
$C_{indir}(E_2) =$	$\{(\forall aniAccept.\forall transport.Cage, H_1, \leq 0aniAccept),$ $(\forall aniAccept.Chien, H_2, \leq 0aniAccept),$ $(\forall aniAccept.Chien, T_2, \leq 0aniAccept),$ $(\forall aniAccept. \leq 10kg, H_2, \leq 0aniAccept),$ $(\forall aniAccept. \leq 10kg, T_3, \leq 0aniAccept)\}$
$C_{egal}(E_2) =$	$\{(T_2, Vol),$ $(T_3, Vol),$ $(T_2, \forall lieuArr.Ville),$ $(T_3, \forall lieuArr.Ville),$ $(H_1, Heberg),$ $(H_2, Heberg),$ $(H_1, \leq 2etoile),$ $(H_1, \forall equip.Piscine),$ $(T_2, \forall tv.EcranPlat),$ $(H_2, \forall tv.EcranPlat),$ $(T_3, \forall escale.\forall duree. \leq 1heure),$ $(H_1H_2, \forall tv.EcranPlat),$ $(H_1H_2, \leq 0aniAccept),$ $(T_2T_3H_1, \leq 0aniAccept)\}$

TAB. 4.8 – Les ensembles $C_{dir}(E_2)$, $C_{indir}(E_2)$ et $C_{egal}(E_2)$ de l'exemple courant.

On peut remarquer qu'on a ici uniquement des singletons pour ensembles H issus des triplets de $C_{indir}(E_2)$[19]. Après fusion on obtient les ensembles de S_i suivants :
$\{H_1, H_2\}$, $\{T_2, H_1, H_2\}$, $\{T_3, H_1, H_2\}$ et $\{T_2, T_3, H_1\}$. Après minimisation, on obtient :
$\{H_1, H_2\}$ et $\{T_2, T_3, H_1\}$.
On a donc montré que $H_1 \sqcap H_2$ et $T_2 \sqcap T_3 \sqcap H_1$ sont les plus petites conjonctions Y de S_i, sous-ensembles de E_2, qui couvrent indirectement $c_Q =\leq 0aniAccept$ avec exactement les mêmes clauses que E_2 (i.e. c_{13}, c_{14} et c_{15}). On ajoute donc à $C_{egal}(E_2)$ les couples $(\{H_1, H_2\}, \leq 0aniAccept)$ et $(\{T_2, T_3, H_1\}, \leq 0aniAccept)$. ◦

Etape 2 : calcul de $R_{eq}(E^*)$ et de I_{rest}

On rappelle que, dans cette étape, on cherche à calculer les $R_{eq}(E^*)$ en combinant tous les Y associés à une même clause c_Q dans les couples de $C_{egal}(E^*)$ précédemment obtenus. En ne gardant que les combinaisons de Y minimales par rapport à l'inclusion, on obtient les ensembles X qui sont les plus petits sous-ensembles de E^* de rest équivalent. Il n'est pas difficile d'exprimer cette étape sous la forme d'un problème de recherche de transversaux minimaux dans un hypergraphe construit à partir de $C_{egal}(E^*)$. C'est l'objet du théorème 4.4.2.

Théorème 4.4.2
Soit $\mathcal{T}$ une $\mathcal{ALN}$-terminologie et Q une $\mathcal{ALN}$-description de concept. Soient C_{dir}, C_{indir}, S_{rest} et $C_{egal}(E^*)$ les ensembles définis précédemment, avec E^* un élément de S_{rest}.
Soit $\mathcal{H}_{E^*}$ l'hypergraphe défini ainsi :
- l'ensemble des sommets est l'ensemble des Y des couples (Y, c_Q) de $C_{egal}(E^*)$:
$\{Y \mid \exists (Y, c_Q) \in C_{egal}(E^*)\}$,
- l'ensemble des arêtes est l'ensemble des c_Q des couples (Y, c_Q) de $C_{egal}(E^*)$:
$\{c_Q \mid \exists (Y, c_Q) \in C_{egal}(E^*)\}$ *et*
- les arêtes sont remplies ainsi : $Y \in c_Q \Leftrightarrow (Y, c_Q) \in C_{egal}(E^*)$.

L'ensemble $R_{eq}(E^*)$ défini par :
$R_{eq}(E^*) = \{X \subseteq E^* \mid X$ *est minimal par rapport à l'inclusion et* $Rest_{E^*}(Q) \equiv Rest_X(Q)\}$
peut être calculé de la manière suivante :

$$R_{eq}(E^*) = Tr(\mathcal{H}_{E^*})$$

Dans l'expression précédente, on sous-entend que l'on fusionne les ensembles de S_i constituants les transversaux minimaux, et que l'on ne garde que les ensembles fusionnés minimaux par inclusion.

La démonstration détaillée se trouve dans l'annexe A.10 page 162.

Illustrons ce théorème sur l'exemple courant.

Exemple courant 12
Poursuivons l'étude de E_2. D'après les couples de $C_{egal}(E_2)$, l'hypergraphe $\mathcal{H}_{E_2}$ est le suivant :
– l'ensemble des sommets est : $\{\{T_2\}, \{T_3\}, \{H_1\}, \{H_2\}, \{H_1, H_2\}, \{T_2, T_3, H_1\}\}$
– l'ensemble des arêtes est :
  – $Vol = \{\{T_2\}, \{T_3\}\}$

[19] D'une manière générale, on peut observer que ces ensembles H des triplets de $C_{indir}(E_2)$ ne sont pas des singletons uniquement dans le cas où une clause d'exclusion (issue d'une inconsistance implicite entre au moins deux S_i) couvre indirectement une clause de Q.

– $\forall lieuArr.Ville = \{\{T_2\}, \{T_3\}\}$
– $Heberg = \{\{H_1\}, \{H_2\}\}$
– $\leq 2etoile = \{\{H_1\}\}$
– $\forall equip.Piscine = \{\{H_1\}\}$
– $\forall tv.EcranPlat = \{\{T_2\}, \{H_2\}, \{H_1, H_2\}\}$
– $\forall escale.\forall duree. \leq 1h = \{\{T_3\}\}$
– $\leq 0aniAccept = \{\{H_1, H_2\}, \{T_2, T_3, H_1\}\}$

Les transversaux minimaux de $\mathcal{H}_{E_2}$ sont
– $\{\{T_3\}, \{H_1\}, \{T_2\}, \{T_2, T_3, H_1\}\}$,
– $\{\{T_3\}, \{H_1\}, \{H_2\}, \{T_2, T_3, H_1\}\}$ et
– $\{\{T_3\}, \{H_1\}, \{H_1, H_2\}\}$.

Après fusion on obtient :
– $\{T_2, T_3, H_1\}$,
– $\{T_2, T_3, H_1, H_2\}$ et
– $\{T_3, H_1, H_2\}$.

Après minimisation, on a $R_{eq}(E_2)$= $\{\{T_2, T_3, H_1\}, \{T_3, H_1, H_2\}\}$.
Si l'on en a besoin, le rest de E_2 est calculé sous la forme d'une description à partir du tableau 4.7 page 80. Les clauses de Q qui sont couvertes directement par E_2 n'appartiennent à aucune description du rest. Celles qui le sont indirectement induisent autant de descriptions dans le rest que le nombre de clauses qui les couvrent indirectement. C'est l'approximation faible de leur négation qui compose les descriptions du rest. Celles qui ne sont pas couvertes appartiennent à toutes les descriptions du rest. Ainsi, on a :

$$\begin{aligned} Rest_{E_2}(Q) = \quad & \{\{c_4 \sqcap \forall aniAccept \forall transport.\neg Cage \sqcap \forall aniAccept. \geq 1transport\}, \\ & \{c_4 \sqcap \forall aniAccept.\neg Chien\}, \\ & \{c_4 \sqcap \forall aniAccept. \geq 11kg\}\} \end{aligned}$$

∘

Une fois les $R_{eq}(E^*)$ calculés, leur union constitue l'ensemble I_{rest}. De par la nature des $R_{eq}(E^*)$, il est clair que I_{rest} contient tous les plus petits ensembles de S_i de rest maximal. I_{rest} est la deuxième borne inférieure de l'espace de recherche.

Comme on l'a dit dans la section 4.3.4, une fois S_{rest} et I_{rest} calculés, il reste à parcourir tous les sous-ensembles des E^* de S_{rest} qui sont aussi sur-ensembles d'un X de I_{rest}. En ne gardant que les ensembles de S_i de miss maximal et minimaux par inclusion, on obtient les meilleures couvertures.

On pourrait penser que parcourir tous les sous-ensembles d'un E^* rend inutile le calcul par le théorème 4.4.2 de tous ses plus petits sous-ensembles X de reste équivalent, c'est-à-dire de $R_{eq}(E^*)$. En effet, pour déterminer, pendant le parcours, le rest de l'ensemble de S_i courant, on pourrait tester son rest avec celui E^* (grâce au lemme 4.3.5), au lieu de tester si l'ensemble de S_i courant contient un X de I_{rest}. Cela pourrait certes être une démarche possible, cependant il n'est pas sûr que construire I_{rest} et tester l'équivalence des rest (avec le lemme 4.3.5) pour chaque sous-ensemble de E^* soit moins coûteux que tester l'inclusion des X dans chaque sous-ensemble de E^*. De plus, dans certaines applications, il peut être intéressant de rechercher les plus petites couvertures de rest maximal (quand par exemple on cherche les meilleures couvertures mais sans s'intéresser à la maximisation du miss). Dans ce cas, les plus petites couvertures de rest maximal sont données directement par I_{rest}.

Exemple courant 13
Pour terminer l'exemple courant, nous montrons, toujours avec le cas de E_2, comment le fait de connaître les clauses de E_2 permet de calculer le miss correspondant. Le principe est le même

que pour le calcul du rest : pour chaque clause c de E_2, on regarde si c est couverte ou non par Q et si c est couverte indirectement par Q on fait la liste des clauses de Q qui couvrent indirectement c. Ensuite, le miss contient les clauses de E_2 non couvertes et les clauses issues des clauses de Q couvrant indirectement les clauses de E_2. Pour E_2, on a :
$Miss_{E_2}(Q) = \{\{c_3 \sqcap c_7 \sqcap c_8 \sqcap c_{11} \sqcap c_{21} \sqcap c_{25} \sqcap c_{31} \sqcap c_{32} \sqcap c_{33} \sqcap \forall tv.\neg EcranPlat\}\}$ ∘

4.4.5 L'algorithme *computeALNBCov*

Commençons par récapituler dans le tableau 4.9 tous les ensembles qui ont été définis précédemment.

Nous donnons maintenant l'algorithme 4 *computeALNBCov* dans une version synthétique qui met en évidence les calculs des bornes de l'espace de recherche. Il fait appel aux fonctions suivantes :

- $CalcIncons()$ qui calcule tous les cas d'inconsistance explicite et implicite et les ensembles S_{cons}, C_{dir} et C_{indir},
- $MaxRest()$ qui calcule de calculer les plus grandes et plus petites couvertures de rest maximal (i.e. respectivement les ensembles S_{rest} et I_{rest}), et
- $MaxMiss()$ qui calcule les meilleures couvertures en parcourant les ensembles de S_i de rest maximal.

Ces trois fonctions sont détaillées dans le tableau synoptique 4.10, page 87, qui résume l'approche dans son ensemble : la première colonne positionne les fonctions précédentes dans le processus, la seconde rappelle les grandes étapes de l'algorithme, la troisième donne la référence au théorème ou au lemme qui énonce la propriété utilisée pour caractériser les éléments importants de l'étape courante, la quatrième donne la référence au lemme ou au théorème utilisé pour les calculs de l'étape courante et la dernière précise les sous-étapes qui décomposent l'étape courante, c'est-à-dire détaille cette étape de l'algorithme.

Algorithme 4 *computeALNBCov*, algorithme de calcul des meilleures couvertures d'un concept en utilisant une terminologie, pour le langage $\mathcal{ALN}$.

Entrée(s) : Une instance $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$ du problème de découverte des meilleures couvertures d'un concept pour le langage $\mathcal{ALN}$, i.e. une terminologie dépliée $\mathcal{T}$ de définitions de concepts S_i normalisées, et une descriptions Q normalisée. On pose :
- $\mathcal{S}_{\mathcal{T}}$ est l'ensemble des S_i de $\mathcal{T}$,
- $\mathcal{S}_{\mathcal{T}_Q}$ est l'union de $\mathcal{S}_{\mathcal{T}}$ et de Q,
- $\mathcal{C}_Q = \bigcup_{S_i \in \mathcal{S}_{\mathcal{T}_Q}} \widehat{\mathcal{G}}^{\#}_{S_i}$ et $\mathcal{C} = \bigcup_{S_i \in \mathcal{S}_{\mathcal{T}}} \widehat{\mathcal{G}}^{\#}_{S_i}$.

Sortie(s) : L'ensemble $bcov(\mathcal{T}, Q)$ des meilleures couvertures $\mathcal{ALN}$ de Q en utilisant $\mathcal{T}$.

1: Calcul de la borne supérieure S_{cons} et de C_{dir} et C_{indir}.
$(S_{cons}, C_{dir}, C_{indir}) := CalcIncons(\mathcal{S}_{\mathcal{T}_Q}, \mathcal{C}_Q)$;
2: Calcul de la borne supérieure S_{couv}.
$S_{couv} := \{E \in S_{cons} \mid \exists (c, H, c_Q) \in C_{dir} \cup C_{indir} \mid H \subseteq E\}$;
3: Calcul de la borne supérieure S_{rest} et de la borne inférieure I_{rest}.
$(S_{rest}, I_{rest}) := MaxRest(C_{dir}, C_{indir}, S_{couv}, \mathcal{S}_{\mathcal{T}}, \mathcal{C})$;
4: Calcul des meilleures couvertures $\mathcal{ALN}$ de Q en utilisant $\mathcal{T}$.
$bcov(\mathcal{T}, Q) := MaxMiss(S_{rest}, I_{rest}, \mathcal{S}_{\mathcal{T}}, \mathcal{C})$;

Remarquons que dans l'algorithme 4 et le tableau synoptique 4.10, la borne inférieure I_{couv}

Nom de l'ensemble	Nom de ses éléments	Description du contenu
Inc_Q	$E_\perp$	Tous les plus petits ensembles de S_i (Q inclus) inconsistants par conjonction.
Inc	E'	Tous les plus petits ensembles de S_i (Q non inclus) inconsistants avec Q par conjonction.
S_{cons}	E	Tous les plus grands ensemble de S_i consistants avec Q (par conjonction). Borne supérieure 1.
C_{dir}	(c, H, c_Q)	Tous les triplets (c, H, c_Q) où c est une clause de H qui couvre directement c_Q une clause de Q, avec H minimal par rapport à l'inclusion.
C_{indir}	(c, H, c_Q)	Tous les triplets (c, H, c_Q) où c est une clause de H qui couvre indirectement c_Q une clause de Q, avec H minimal par rapport à l'inclusion.
I_{couv}	H	Tous les plus petits ensemble de S_i consistants avec Q et couvrant (directement ou indirectement) au moins une clause de Q (i.e. toutes les plus petites couvertures de Q). Borne inférieure 1.
S_{couv}	E'	Tous les plus grands ensemble de S_i consistants avec Q et couvrant (directement ou indirectement) au moins une clause de Q (i.e. toutes les plus grandes couvertures de Q). Borne supérieure 2.
$cov(\mathcal{T}, Q)$	E_{couv}	Tous les ensembles de S_i qui sont des couvertures de Q en utilisant $\mathcal{T}$.
S_{rest}	E^*	Toutes les plus grandes couvertures de Q de rest maximal. Borne supérieure 3.
$C_{dir}(E^*)$	(c, H, c_Q)	Avec $E^* \in S_{rest}$: Tous les triplets (c, H, c_Q) où c est une clause de H qui couvre directement c_Q une clause de Q, avec H minimal par rapport à l'inclusion et $H \subseteq E^*$.
$C_{indir}(E^*)$	(c, H, c_Q)	Avec $E^* \in S_{rest}$: Tous les triplets (c, H, c_Q) où c est une clause de H qui couvre indirectement c_Q une clause de Q, avec H minimal par rapport à l'inclusion et $H \subseteq E^*$.
$C_{egal}(E^*)$	(Y, c_Q)	Avec $E^* \in S_{rest}$: Tous les couples (Y, c_Q) où c_Q est une clause de Q couverte de la même façon (directement ou indirectement et avec les mêmes clauses) par E^* et par Y, avec Y minimal par rapport à l'inclusion et $Y \subseteq E^*$.
$R_{eq}(E^*)$	X	Avec $E^* \in S_{rest}$: Tous les plus petits ensembles de S_i sous-ensembles de E^* de rest équivalent.
I_{rest}	X	Toutes les plus petites couvertures de Q de rest maximal. Borne inférieure 2.
$cov_{rest}(\mathcal{T}, Q)$	E_{rest}	Tous les ensembles de S_i qui sont des couvertures de Q en utilisant $\mathcal{T}$ de rest maximal.
$bcov(\mathcal{T}, Q)$	E_{best}	Tous les ensembles de S_i qui sont des meilleures couvertures de Q en utilisant $\mathcal{T}$.

TAB. 4.9 – Récapitulatif de tous les ensembles calculés au cours de la découverte des meilleures couvertures dans $\mathcal{ALN}$.

	Etapes de l'algorithme	**Car.**	**Calc.**	**Algorithme *computeALNBCov***
CalcIncons()	• **Inconsistances explicites** : calcul de $S_{cons} = \{E \subseteq \mathcal{S}_\mathcal{T} \mid E \sqcap Q \not\equiv \bot$ et E max p/r à $\subseteq\}$	lemme 4.2.2 lemme 4.3.1	théo. 4.4.1	• $Inc_Q = Min_\subseteq \left\{ X \mid X \in \bigcup_{c' \in \mathcal{C}_Q} (Tr(\mathcal{H}_{c'}, \mathcal{S}_{\mathcal{T}_\mathcal{Q}})) \right\}$ avec $\mathcal{C}_Q = \bigcup_{S_i \in \mathcal{S}_{\mathcal{T}_\mathcal{Q}}} \widehat{\mathcal{G}}^{\#}_{S_i}$ et $\mathcal{S}_{\mathcal{T}_\mathcal{Q}} = \mathcal{S}_\mathcal{T} \cup \{Q\}$
				• $Inc = Min_\subseteq \{E_\bot \setminus \{Q\} \mid E_\bot \in Inc_Q\}$
			lemme 4.4.1	• $S_{cons} = \{\mathcal{S}_\mathcal{T} \setminus X \mid X \in Tr(\mathcal{H}_{Inc})\}$
	• **Inconsistances implicites** : calcul de $C_{dir} = \{(c, H, c_Q) \mid H \subseteq \mathcal{S}_\mathcal{T}$ et H min. p/r à $\subseteq$ et $c \in \widehat{\mathcal{G}}^{\#}_{\sqcap_{S_i \in H} S_i}$ et $c_Q \in \widehat{\mathcal{G}}^{\#}_Q$ et c couvre dir. $c_Q\}$ $C_{indir} = \{(c, H, c_Q) \mid H \subseteq \mathcal{S}_\mathcal{T}$ et H min. p/r à $\subseteq$ et $c \in \widehat{\mathcal{G}}^{\#}_{\sqcap_{S_i \in H} S_i}$ et $c_Q \in \widehat{\mathcal{G}}^{\#}_Q$ et c couvre indir. $c_Q\}$	lemme 4.3.2	lemme 4.2.2 et théo. 4.4.1	• **Pendant le calcul de Inc_Q :** – Faire les calculs de Tr avec l'algo. 2 (ajout d'une arête par itération). – A partir de la troisième itération, les non persistants X correspondent à des inconsistances implicites aboutissant à une clause d'exclusion c_X dans l'ensemble de S_i correspondant à X. • Réunir tous les (X, c_X) dans un ensemble $\mathcal{X}$. • Supprimer de $\mathcal{X}$ les couples (X, c_X) (i) ayant Q dans X et (ii) tels que $\exists (X, c'_X) \in \mathcal{X}$ avec $c'_X \sqsubset c_X$. • On a C_{dir} et C_{indir} en comparant les clauses des S_i et des couples de $\mathcal{X}$ avec celles de Q.
	• **Couvertures de Q** : calcul de $S_{couv} = \{E \in S_{cons} \mid \exists (c, H, c_Q) \in C_{dir} \cup C_{indir} \mid E \supseteq H\}$ $cov(\mathcal{T}, Q) =$ $\{E_{couv}$ couvertures de $Q\}$	théo. 4.3.1		• Comparaison des E de S_{cons} et des H de $C_{dir} \cup C_{indir}$: – $S_{couv} := \emptyset$ – $\forall (E, H) \in S_{cons} \times (C_{dir} \cup C_{indir})$, si $H \subseteq E$ alors $S_{couv} := S_{couv} \cup \{E\}$ • $cov(\mathcal{T}, Q) = \{E_{couv} \subseteq \mathcal{S}_\mathcal{T} \mid \exists E' \in S_{couv} \mid E_{couv} \subseteq E'$ et $\exists (c, H, c_Q) \in C_{dir} \cup C_{indir} \mid E_{couv} \supseteq H\}$
MaxRest()	• **Couv. de Q de rest max** : calcul de $S_{rest} = \{E' \in S_{couv} \mid rest_{E'}(Q)$ max. p/r à $\sqsubseteq\}$	lemme 4.3.3 lemme 4.3.4		• Comparaison des éléments de S_{couv} 2 à 2 : – $S_{rest} := \emptyset$ – $\forall (E'_1, E'_2) \in S_{couv} \times S_{couv}$, – si $Rest_{E'_1}(Q) \sqsubset Rest_{E'_2}(Q)$ (lemme 4.3.3) si $E'_1 \in S_{rest}$ alors $S_{rest} := (S_{rest} \setminus \{E'_1\}) \cup \{E'_2\}$ sinon $S_{rest} := S_{rest} \cup \{E'_2\}$
	$\forall E^* \in S_{rest}$: $R_{eq}(E^*) = \{X \subseteq E^* \mid X$ min. p/r à $\sqsubseteq$ et $Rest_{E^*}(Q) \equiv Rest_X(Q)\}$	lemme 4.3.5	lemme 4.4.2	• $C_{dir}(E^*) = \{(c, H, c_Q) \in C_{dir} \mid H \subseteq E^*\}$ • $C_{indir}(E^*) = \{(c, H, c_Q) \in C_{indir} \mid c$ min. p/r à $\sqsubseteq \mid H \subseteq E^*\}$ • $C_{egal}(E^*) = \{(Y, c_Q) \mid \exists (c, H, c_Q) \in C_{dir}(E^*)$ et $H = Y\}$ $\cup \{(Y, c_Q) \mid \exists (c', H', c_Q) \in C_{indir}(E^*)$ et $Y \in Min_\subseteq (\bigtimes_{\widetilde{c} \in \widehat{\mathcal{G}}^{\#}_{E^*}} \{\widetilde{H} \mid (\widetilde{c}, \widetilde{H}, c_Q) \in C_{indir}(E^*)\})\}$
	$I_{rest} = \bigcup_{E^* \in S_{rest}} R_{eq}(E^*)$		théo. 4.4.2	• $\forall E^* \in S_{rest}$, $R_{eq}(E^*) = Tr(\mathcal{H}_{E^*})$
	$cov_{rest}(\mathcal{T}, Q) = \{E_{rest} \in cov(\mathcal{T}, Q)$ de rest max$\}$	théo. 4.4.2		• $cov_{rest}(\mathcal{T}, Q) = \{E_{rest} \subseteq \mathcal{S}_\mathcal{T} \mid \exists E^* \in S_{rest} \mid E_{rest} \subseteq E^*$ et $\exists X \in I_{rest} \mid E_{rest} \supseteq X\}$
MaxMiss()	• **Meilleures couv. de Q** : calcul de $bcov(\mathcal{T}, Q) =$ {meilleures couvertures de Q}			• Parcours de tous les sous-ensembles des E^* de S_{rest} qui sont aussi sur-ensembles d'un X de I_{rest} : on ne garde que ceux de miss max. • Tri par rapport à $\subseteq$ des ensembles de S_i trouvés à l'étape précédente : on garde les min.

TAB. 4.10 – Tableau synoptique des grandes étapes de la découverte des meilleures couvertures $\mathcal{ALN}$ de Q en utilisant $\mathcal{T}$ et de l'algorithme *computeALNBCov* correspondant.

n'est pas évoquée. En effet, I_{couv} n'a pas besoin d'être explicitement calculée. Seuls C_{dir} et C_{indir} doivent l'être puisque la seconde borne inférieure I_{rest} est calculée à partir d'eux.

4.5 Complexité des meilleures couvertures

Nous nous intéressons maintenant à la complexité du problème $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$, puis à celle de l'algorithme *compute*$\mathcal{ALN}$*BCov*.

4.5.1 Complexité du problème $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$

Dans cette étude de complexité de $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$, nous réduisons polynomialement $\mathcal{BCOV}(\mathcal{T}, Q)$ dans $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$. Ainsi, $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$ est au moins aussi difficile que $\mathcal{BCOV}(\mathcal{T}, Q)$. $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$ est donc NP-Difficile.

Plus précisément, nous montrons que la définition des meilleures couvertures pour $\mathcal{FL}_0$ considéré comme un langage à subsomption structurelle coïncide avec la définition des meilleures couvertures pour $\mathcal{FL}_0$ considéré comme un sous-langage d'$\mathcal{ALN}$. Ainsi, résoudre $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$ pour $\mathcal{T}$ et Q exprimés avec $\mathcal{FL}_0$ revient à chercher un ensemble de meilleures couvertures contenant les meilleures couvertures obtenues par résolution de $\mathcal{BCOV}(\mathcal{T}, Q)$ avec les mêmes $\mathcal{T}$ et Q. Comme $\mathcal{BCOV}(\mathcal{T}, Q)$ est NP-Difficile pour les langages à subsomption structurelle, alors $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$ est aussi NP-Difficile.

Théorème 4.5.1 (Complexité de $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$)
Soient $\mathcal{T}$ une $\mathcal{ALN}$-terminologie et Q une $\mathcal{ALN}$-description.
$\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$ est un problème NP-Difficile.

La démonstration de ce théorème est donnée dans l'annexe A.11 page 163. L'idée est de démontrer les points intermédiaires suivants :

- La découverte des couvertures de meilleur rest dans $\mathcal{FL}_0$ est le même problème quand $\mathcal{FL}_0$ est considéré comme un langage à subsomption structurelle ou comme un sous-langage d'$\mathcal{ALN}$.
- Pour $\mathcal{FL}_0$, les solutions de $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$ (les couvertures ayant meilleurs rest et miss et minimales par rapport à l'inclusion) englobent les solutions de $\mathcal{BCOV}(\mathcal{T}, Q)$.

4.5.2 Complexité de l'algorithme *compute*$\mathcal{ALN}$*BCov*

Le tableau 4.11 résume les complexités de chaque étape de *compute*$\mathcal{ALN}$*BCov* en donnant les bornes supérieures des cardinalités de tous les ensembles calculés, ainsi que les bornes supérieures du temps de calcul nécessaire à leur obtention.

Ces résultats sont des bornes supérieures théoriques : leur rôle est plus de pouvoir comparer les complexités des différentes étapes de *compute*$\mathcal{ALN}$*BCov* entre elles afin d'identifier les plus coûteuses, que de donner une mesure représentative de la complexité d'exécution de l'algorithme implémenté. Ceci est dû aux faits que ce sont des résultats de complexité au pire, qu'ils sont basés sur une borne supérieure peu précise de la complexité du problème des transversaux minimaux, et que l'enchaînement des différentes étapes dans *compute*$\mathcal{ALN}$*BCov* implique que les imprécisions successives se cumulent. En ce qui concerne ce cumul, nous n'avons pas réussi à trouver de meilleures bornes supérieures que 2^v, valeur évidente, pour les cardinalités des ensembles des couvertures, des couvertures de rest maximal ou des meilleures couvertures. De plus, c'est ce même problème du cumul des imprécisions qui justifie que nous ne donnions pas un résultat global de complexité pour *compute*$\mathcal{ALN}$*BCov*.

Grandes étapes de *compute$\mathcal{ALN}$BCov*	**Bornes supérieures des cardinalités**	**Complexités en temps au pire**
• **Inconsistances explicites** : calcul de S_{cons}	• $\|Inc_Q\| \leq (vl+N)(v+1)^{(p+1)}$	• $\mathcal{O}(l^2N^2(v+1)^{2p+4})$
	• $\|Inc\| \leq (vl+N)(v+1)^{p+1}$	• $\mathcal{O}(l^2N^2(v+1)^{2p+5})$
	• $\|S_{cons}\| \leq v^{\mathcal{I}_e}$ avec • $\mathcal{I}_e \leq lN(v+1)^{p+2}$	• $\mathcal{O}(\mathcal{I}_e v^{\mathcal{I}_e+1})$
• **Inconsistances implicites** : calcul de C_{dir} et C_{indir}	• $\|C_{dir}\| \leq N(vl+\mathcal{I}_i) \leq 2Nlv^{p+2}$ • $\|C_{indir}\| \leq N(vl+\mathcal{I}_i) \leq 2Nlv^{p+2}$ • $\mathcal{I}_i \leq lv^{p+2}$	• $\mathcal{O}(l^2N(v+1)^{2p+4})$
• **Couvertures de Q en util. $\mathcal{T}$** : calcul de S_{couv} et $cov(\mathcal{T},Q)$	• $\|S_{couv}\| \leq v^{\mathcal{I}_e}$	• $\mathcal{O}(l^2Nv^{\mathcal{I}_e+p+4})$
	• $\|cov(\mathcal{T},Q)\| \leq 2^v$	• $\mathcal{O}(lNv^{\mathcal{I}_e+p+3} * 2^v)$
• **Couvertures de Q de rest max** : calcul de S_{rest} et pour tout $E^* \in S_{rest}$ des $R_{eq}(E^*)$ (et donc de I_{rest}), ainsi que $cov_{rest}(\mathcal{T},Q)$	• $\|S_{rest}\| \leq v^{\mathcal{I}_e}$	• $\mathcal{O}(Nl^3p^4v^{2\mathcal{I}_e+p+4})$
	Pour tout E^* *in* S_{rest} : • $\|C_{dir}(E^*)\| \leq N(vl+\mathcal{I}_i)$ • $\|C_{indir}(E^*)\| \leq N(vl+\mathcal{I}_i)$ • $\|C_{egal}(E^*)\| \leq N2^v$	Pour tout E^* *in* S_{rest} : • $\mathcal{O}(l^2v^{p+4})$ • $\mathcal{O}(l^3p^3v^{p+4})$ • $\mathcal{O}((2lv^{p+2})^{2vlp+3})$
	Pour tout E^* *in* S_{rest} : • $\|R_{eq}(E^*)\| \leq 2^v$	Pour tout E^* *in* S_{rest} : • $\mathcal{O}(vN2^{vN})$
	• $\|cov_{rest}(\mathcal{T},Q)\| \leq 2^v$	• $\mathcal{O}(v^{\mathcal{I}_e+1} * 2^{2v})$
• **Meilleures couvertures de Q** : calcul des couvertures de rest et miss max calcul de $bcov(\mathcal{T},Q)$	• $\|bcov(\mathcal{T},Q)\| \leq 2^v$	• calcul des miss $\mathcal{O}(l^2p^2v^{p+4}N^{2vl(p+1)})$ • comparaison des miss $\mathcal{O}(l^2p^3v^2N^{2vl(p+1)})$ • minimisation p/r à $\subseteq$ $\mathcal{O}(v2^{2v})$

Avec :
- $v = |\mathcal{S}_\mathcal{T}|$, i.e. v est le nombre de concepts définis S_i de $\mathcal{T}$,
- $p = Max(prof(c)|c \in \widehat{\mathcal{G}}^\#_{S_1} \cup ... \cup \widehat{\mathcal{G}}^\#_{S_v} \cup \widehat{\mathcal{G}}^\#_Q)$, i.e. p est la profondeur maximale d'une clause présente dans un $\widehat{\mathcal{G}}^\#_{S_i}$ ou dans $\widehat{\mathcal{G}}^\#_Q$,
- $l = Max_i(|\widehat{\mathcal{G}}^\#_{S_i}|)$, i.e. l est le nombre de clauses maximal d'un $\widehat{\mathcal{G}}^\#_{S_i}$ et
- $N = |\widehat{\mathcal{G}}^\#_Q|$, i.e. N est le nombre de clauses de $\widehat{\mathcal{G}}^\#_Q$.

TAB. 4.11 – Récapitulatif des complexités des grandes étapes de *compute$\mathcal{ALN}$BCov*.

Dans les meilleures couvertures, l'optimisation du miss correspond à l'optimisation d'un coût associé aux réécritures. Ce coût, et donc le miss, peut être défini de diverses manières suivant le contexte de l'utilisation des meilleures couvertures. Nous commenterons donc assez précisément les résultats de complexité concernant la découverte des couvertures de rest maximal, mais plus succintement ceux liés à la maximisation du miss.

En résumé, la recherche est généralement polynomiale en le nombre N de clauses de Q, en le nombre maximal l de clauses dans un S_i et en le nombre v de S_i dans la terminologie $\mathcal{T}$, et est exponentielle en la profondeur maximale p d'une clause d'un S_i ou de Q (c'est le cas des calculs de Inc_Q, Inc, $\mathcal{I}_e$, C_{indir}, $\mathcal{I}_i$, les $C_{dir}(E^*)$, les $C_{indir}(E^*)$ et les $C_{egal}(E^*)$). Certaines étapes sont plus complexes car exponentielles en le nombre $\mathcal{I}_e$ d'inconsistances explicites (lui-même obtenu en temps polynomial en N, v et l et exponentiel en p), ou en N, l et v (c'est le cas des calculs de S_{cons}, S_{couv}, S_{rest}, les $C_{egal}(E^*)$, les $R_{eq}(E^*)$ (et donc I_{rest}), $cov(\mathcal{T}, Q)$ et $cov_{rest}(\mathcal{T}, Q)$).

Les complexités exponentielles en $\mathcal{I}_e$ s'expliquent par le grand nombre de configurations d'inconsistances possibles et, de ce fait, ne peuvent être évitées. Un point positif, cependant, est que la découverte des inconsistances explicites est effectuée a priori, c'est-à-dire en début de réécriture, et non a posteriori, c'est-à-dire en fin d'algorithme dans une phase de vérification des réécritures. Cette recherche a priori permet de ne découvrir les inconsistances qu'une fois pour toutes. Enfin on peut imaginer, par exemple dans le cas de l'application des meilleures couvertures à la découverte dynamique de services web, que les cas concrets d'inconsistance explicite entre S_i (entre services) sont rares par rapport au nombre total de combinaisons de S_i possibles.

Le calcul des ensembles $C_{egal}(E^*)$ et des ensembles $R_{eq}(E^*)$ est exponentiel en fonction de p, mais aussi N et l et surtout v : dans la découverte dynamique de services, cette valeur v peut être grande (au moins quelques milliers, voire plus). Ces calculs étant nécessaires à la maximisation du rest, essentielle à la flexibilité de notre réécriture, il n'est pas surprenant que la contrepartie de cette flexibilité soit une grande complexité dans les mécanismes qui l'assurent. En regardant plus en détails les sources de cette complexité, on voit qu'elle tient essentiellement en la manipulation d'hypergraphes dont les sommets sont des ensembles de S_i, et non des S_i séparés : le calcul des $C_{egal}(E^*)$ est la construction de ces hypergraphes, et celui des $R_{eq}(E^*)$ est la recherche de leurs transversaux minimaux. Comme les sommets sont des ensembles de S_i et non des S_i séparés, le nombre de sommets potentiels est exponentiellement plus grand que le nombre v de S_i. Dès lors, les étapes de construction et de recherche de transversaux minimaux sont inévitablement plus complexes. D'un point de vue théorique, cette complexité vient du fait que les clauses d'exclusion (issues d'inconsistances implicites) peuvent couvrir indirectement les clauses de Q. C'est ce qui rend difficile la maximisation du rest. On rappelle que deux couvertures ont un rest équivalent si et seulement si elle couvre exactement les mêmes clauses de Q et les clauses de Q couvertes indirectement doivent l'être par exactement les mêmes clauses (voir le lemme 4.3.5 page 67). Sachant qu'une clause de Q couverte indirectement peut l'être par une clause d'exclusion issue d'une ou plusieurs inconsistances implicites différentes pouvant survenir dans plusieurs ensembles de S_i, la combinatoire est très importante, et donc la complexité aussi.

L'énumération des couvertures et des couvertures de rest maximal est exponentielle en fonction de $\mathcal{I}_e$ et v. Ceci reflète découle des résultats précédents et reflète les imprécisions cumulées qu'ils impliquent.

Par rapport aux résultats précédents, la complexité de la maximisation du miss n'est pas négligeable : maximiser les couvertures par rapport au miss est coûteux puisqu'exponentiel en

fonction de v, l et p, à l'instar des étapes les plus coûteuses de la recherche des couvertures de rest maximal. L'aspect positif de ce résultat est qu'il permet d'avoir une idée du coût des miss qui ne possèdent aucune propriété simplificatrice pour son optimisation, et qui sont donc difficiles à traiter, comme c'est le cas de celui qui est utilisé dans cette thèse. Ce résultat montre aussi l'intérêt du lemme 4.3.3 page 65 permettant de comparer les rest grâce aux clauses des descriptions concernées et sans calculer explicitement ces rest et les comparer deux à deux.

Chapitre 5

Etat de l'art de la réécriture et positionnement des meilleures couvertures

Sommaire

Dans ce chapitre, nous détaillons les instances existantes de la réécriture de concept en utilisant une terminologie et nous positionnons les meilleures couvertures par rapport à elles (section 5.1). Nous comparons ensuite les approches algorithmiques de ces réécritures (section 5.2), en montrant notamment pourquoi l'approche des paniers n'est pas adaptée pour trouver les meilleures couvertures dans le langage $\mathcal{ALN}$.

5.1 Instances existantes de la réécriture de concept

Rappelons d'abord le cadre formel de la réécriture de concept en utilisant une terminologie (voir la définition 2.1.1 page 22 et [7, 8]). Une réécriture E d'une $\mathcal{L}_s$-description de concept Q est une $\mathcal{L}_d$-description exprimée avec des termes d'une $\mathcal{L}_t$-terminologie $\mathcal{T}$. E doit vérifier une relation ρ avec Q, et minimiser un ordre $\preceq$. En dehors des meilleures couvertures, il existe deux autres instances de ce cadre.

La première est la réécriture équivalente de taille minimale, définie dans [7, 8], qui cherche à récrire une description C en les descriptions E équivalentes de taille minimale, où la taille est le nombre d'occurences de noms de concepts et de rôles dans E. Dans cette instance, ρ est donc l'équivalence modulo la terminologie (i.e. $Q \equiv E$ doit être vérifié), et $\preceq$ est l'ordre entre descriptions défini par rapport à leur taille (i.e. $E_1 \preceq E_2 \Leftrightarrow |E_1| \leq |E_2|$). Les trois langages concernés sont les mêmes (i.e. $\mathcal{L}_s = \mathcal{L}_d = \mathcal{L}_t$). L'étude de [7, 8] concerne ainsi $\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$ et $\mathcal{ALC}$[20]. La réécriture minimalement équivalente n'est pas une réécriture totale dans le sens où certaines descriptions peuvent n'être récrites qu'en partie avec des concepts de la terminologie, conservant une partie de la description originale.

[20] $\mathcal{ALC}$ contient les constructeurs d'$\mathcal{ALCN}$ moins les restrictions numériques.

Terminologie $\mathcal{T}$		
$Femme$	$\equiv$	$Personne \sqcap Feminin$
$Homme$	$\equiv$	$Personne \sqcap \neg Femme$
$Mere$	$\equiv$	$Femme \sqcap \exists aEnfant.Personne$
$Pere$	$\equiv$	$Homme \sqcap \exists aEnfant.Personne$
$Parent$	$\equiv$	$Mere \sqcup Pere$
$GrandMere$	$\equiv$	$Femme \sqcap \exists aEnfant.Parent$
$MereGdeFamille$	$\equiv$	$Mere \sqcap \geq 3\ aEnfant$
$MereSansFille$	$\equiv$	$Mere \sqcap \forall aEnfant.\neg Femme$
$Epouse$	$\equiv$	$Femme \sqcap \exists aEpoux.Homme$

TAB. 5.1 – Rappel de la terminologie $\mathcal{T}$ dans le domaine de la famille.

Exemple 20
Nous donnons ci-dessous un exemple de réécriture minimale équivalente utilisant la terminologie du tableau 5.1 concernant le domaine de la famille.
Soit Q la description à couvrir définie comme suit :
$Q \equiv Personne \sqcap Feminin \sqcap \exists aEnfant.(Personne \sqcap Feminin \sqcap \exists aEpoux.(Personne \sqcap \neg Femme))$
La description $E \equiv Mere \sqcap \exists aEnfant.Epouse$ est une réécriture minimalement équivalente de Q. En effet, $Personne \sqcap Feminin \sqcap \exists aEnfant.Personne$ correspond à la définition de $Mere$, et $Personne \sqcap Feminin \sqcap \exists aEpoux.(Personne \sqcap \neg Femme)$ à celle d'$Epouse$. ∘

En termes d'applications, il a été montré dans [7, 8] que cette réécriture minimalement équivalente est très utile dans la gestion de grandes terminologies pour rendre les descriptions complexes plus faciles à lire et à manipuler pour un utilisateur. Les raisonnements non standard, comme par exemple le lcs, qui manipulent des versions dépliées des concepts définis dans la terminologie, produisent souvent des descriptions très longues et difficiles à lire et à manipuler pour un utilisateur. Leur lisibilité est améliorée en les récrivant en des descriptions équivalentes plus petites, obtenues en remplaçant des sous-descriptions par des concepts définis équivalents.

La deuxième instance du cadre formel de la réécriture est la réécriture maximalement contenue. Elle propose de récrire une description Q en des descriptions E subsumées par Q mais maximales par rapport à la subsomption. ρ est donc la subsomption (i.e. $E \sqsubseteq Q$ doit être vérifié), et $\preceq$ la subsomption inverse (i.e. $E_1 \preceq E_2 \Leftrightarrow E_1 \sqsupseteq E_2$). Les réécritures sont totales (i.e. construites uniquement à partir de concepts définis de la terminologie). La réécriture maximalement contenue étudiée dans [12] concerne le langage $\mathcal{L}_d = \{\sqcap, \sqcup\}$ et le langage $\mathcal{ALCNR}$[21] (ou $\mathcal{ALN}$) pour $\mathcal{L}_s$ et $\mathcal{L}_t$. D'autres réécritures maximalement contenues sont étudiées dans [12, 72, 40, 39, 42, 56, 31, 69, 68, 44] et regroupées sous la dénomination de "Datalog rewriting". Elles étendent le cadre de la réécriture à d'autres langages comme notamment les langages relationnels (Datalog ou $\mathcal{CQ}$[22]) ou des langages hybrides de la famille CARIN[23].

Exemple 21
Nous donnons ci-dessous un exemple de réécriture maximalement contenue utilisant la termino-

[21] $\mathcal{ALCNR}$ est la logique de description $\mathcal{ALCN}$ augmentée du constructeur de conjonction pour les rôles défini par $(R \sqcap S)^{\mathcal{I}} = \{(a,b) \in R^{\mathcal{I}} \cap S^{\mathcal{I}}\}$, pour R et S deux rôles et $\mathcal{I}$ une interprétation.
[22] $\mathcal{CQ}$ est le nom du langage regroupant les requêtes conjonctives.
[23] Par exemple, le langage $\mathcal{CQ}$-$\mathcal{ALN}$ est un langage de la famille CARIN qui autorise l'expression de requêtes conjonctives avec prédicats qui sont des $\mathcal{ALN}$-descriptions. Voir [57, 39, 42].

logie du tableau 5.1. Soit la description de concept Q définie comme suit :
$Q \equiv \exists aEpoux \sqcap \exists aEnfant$
La description $E \equiv Mere \sqcap Epouse$ est une réécriture maximalement contenue de Q. En effet, $Mere \sqsubseteq \exists aEnfant$ et $Epouse \sqsubseteq \exists aEpoux$, donc $Mere \sqcap Epouse \sqsubseteq \exists aEpoux \sqcap \exists aEnfant$ et $Mere \sqcap Epouse$ est le plus grand concept par rapport à la subsomption à avoir cette propriété. ◦

En termes d'applications, la réécriture maximalement contenue a été utilisée pour résoudre le problème de réécriture de requêtes en utilisant des vues (en supposant que la requête est la description à récrire et les vues correspondent aux concepts définis de la terminologie). Ce problème se pose notamment dans certaines approches d'intégration de données qui décrivent une architecture de médiation LAV ("Local As View" en anglais [76, 44, 54]) où les sources sont décrites comme des vues sur un schéma global de médiation. Les techniques de réécriture de requêtes en utilisant des vues (comme la réécriture maximalement contenue) permettent alors de résoudre le problème de répondre à des requêtes en utilisant ces vues, c'est-à-dire de chercher toutes les solutions possibles à des requêtes en utilisant les extensions des vues (voir [44] pour un panorama détaillé des travaux relatifs à ce problème).

Le tableau 5.2 ci-après récapitule toutes ces approches et donne les résultats de décidabilité associés. La figure 5.1 donne une vision schématique de toutes ces réécritures par rapport à leur domaine d'application. Dans cette figure, la partie en grisé provient de [44] et situe la réécriture maximalement contenue par rapport au problème de répondre à une requête en utilisant des vues. Pour être le plus exhaustif possible, on situe les raisonnements d'approximation (faible et forte) des logiques de description définis dans [21, 22, 23, 39, 41] : bien que n'étant pas habituellement présentés comme tels, on peut les voir comme des raisonnements de réécriture non totale.

En commentaire du tableau 5.2, nous pouvons dire que la découverte des meilleures couvertures est définie pour des langages un peu moins expressifs que les autres réécritures. En contrepartie, comme évoqué précédemment, elle généralise les réécritures équivalente et maximalement contenue. On peut en effet remarquer qu'une couverture candidate E qui est équivalente à ou subsumée par le concept à couvrir Q implique un rest équivalent à $\top$, et donc un rest minimal en taille et maximal par rapport à la subsomption. Ainsi, les meilleures couvertures découvrent en premier les réécritures équivalentes ou maximalement contenues s'il en existe. Dans le cas d'absence de réécriture équivalente ou maximalement contenue, des meilleures couvertures pourront tout de même être découvertes parmi toutes les autres conjonctions possibles de concepts de la terminologie.

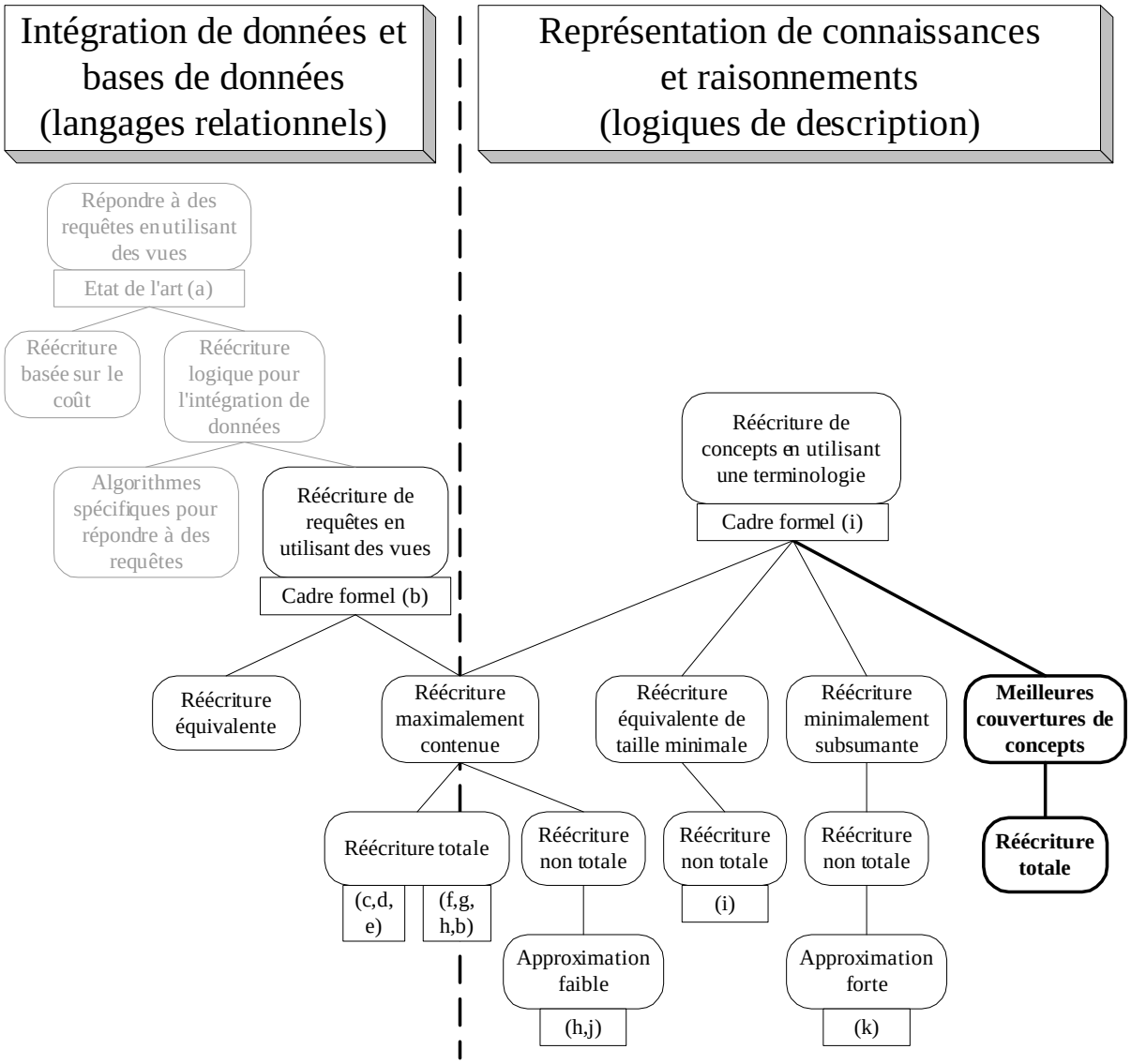
5.2 Approches algorithmiques de la réécriture

L'approche algorithmique utilisée dans la réécriture équivalente de taille minimale [8, 9] consiste à ajouter à la description à récrire des concepts définis dont la définition est une partie de la description à récrire (phase d'extension), puis à enlever toutes les sous-descriptions possibles tout en gardant une description équivalente (phase de réduction). Cette approche est difficilement applicable aux meilleures couvertures car les concepts définis ajoutés en phase d'extension doivent subsumer Q (sinon les descriptions obtenues après réduction peuvent ne pas être équivalentes à la description à récrire). Or les meilleures couvertures peuvent être récrites avec des concepts définis qui ne subsument pas la description à récrire, s'ils impliquent un rest optimal.

	Intitulé et définition	Langages et références	Décidabilité
Réé. min. équiv.	"equivalent rewriting" • ρ : $Q \equiv E$ • $\preceq$: $\|E\|$ minimale • réécriture non totale	$\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$ et $\mathcal{ALC}$ pour $\mathcal{L}_s$, $\mathcal{L}_t$ et $\mathcal{L}_d$. Voir [8, 9]	Le calcul d'une réécriture équivalente est NP-Difficile pour $\mathcal{FL}_0$, $\mathcal{ALN}$ et $\mathcal{ALE}$, et PEspace-Difficile pour $\mathcal{ALC}$. Pour $\mathcal{ALE}$, calculer une ou toutes les réécritures équivalentes est décidable.
Réécriture maximalement contenue	• ρ : E contenue dans Q (i.e. $E \sqsubseteq Q$) • $\preceq$: E maximalement contenue (i.e. max. p/r à $\sqsubseteq$) • réécriture totale	$\mathcal{L}_s$: $\mathcal{ALCNR}$ (ou $\mathcal{ALN}$) $\mathcal{L}_t$: $\mathcal{ALCNR}$ (ou $\mathcal{ALN}$) $\mathcal{L}_d$: $\{\sqcap, \sqcup\}$ Voir [12].	Le calcul d'une réécriture maximalement contenue est décidable.
	"Datalog rewriting"	$\mathcal{L}_s$: CARIN-$\mathcal{ALN}$ $\mathcal{L}_t$: CARIN-$\mathcal{ALN}$ $\mathcal{L}_d$: $\mathcal{CQ}$ Voir [12].	Le calcul d'un ensemble fini de réécritures maximalement contenues est décidable s'il n'y a pas de variable existentielle dans la définition des vues.
		$\mathcal{L}_s$: CARIN-$\mathcal{ALN}$ $\mathcal{L}_t$: $\mathcal{ALN}$ $\mathcal{L}_d$: $\mathcal{CQ}$ Voir [72, 40].	Le calcul de l'ensemble (fini) des réécritures maximalement contenues est décidable.
		$\mathcal{L}_s$: $\mathcal{CQ}$-$\mathcal{ALN}$ (ou $\mathcal{CQ}$-$\mathcal{FL}^{\varepsilon}$ ou $\mathcal{CQ}$-$\mathcal{AL}^{+}$) $\mathcal{L}_t$: $\mathcal{ALN}$ (ou $\mathcal{FLE}$ ou $\mathcal{AL}^{+}$) $\mathcal{L}_d$: $\mathcal{CQ}$ (avec ou sans prédicats d'inégalité) Voir [39, 42].	Le calcul de l'ensemble (fini) des réécritures maximalement contenues est décidable.
		$\mathcal{L}_s$: $\mathcal{CQ}$ avec prédicats de comparaisons $\mathcal{L}_t$: $\mathcal{CQ}$ avec prédicats de comparaisons $\mathcal{L}_d$: DATALOG (union de requêtes conjonctives) Voir [56, 31, 69, 68, 44].	Trouver une réécriture maximalement contenue est décidable.
Meilleures couvertures	$\mathcal{BCOV}(\mathcal{T}, Q)$ • ρ : $Q - lcs(Q, E) \not\equiv Q$ • $\preceq$: $(\|Rest_E(Q)\|, \|Miss_E(Q)\|)$ min. p/r à l'ordre lexicographique et E, comme ensemble de S_i, min. p/r à $\subseteq$. • réécriture totale	$\mathcal{L}_s$ et $\mathcal{L}_t$: un même langage à subsomption structurelle $\mathcal{L}_d$: $\{\sqcap, \sqcup\}$ Voir le chapitre 3.	Le calcul de l'ensemble (fini) des meilleures couvertures est décidable (problème NP-Difficile).
	$\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$ • ρ : $Q - lcs(Q, E) \not\equiv \{Q\}$ • $\preceq$: $Rest_E(Q)$ maximal p/r à $\sqsubseteq$, puis $Miss_E(Q)$ max. p/r à $\sqsubseteq$, et enfin E, comme ensemble de S_i, min. p/r à $\subseteq$. • réécriture totale	$\mathcal{L}_s$: $\mathcal{ALN}$ $\mathcal{L}_t$: $\mathcal{ALN}$ $\mathcal{L}_d$: $\{\sqcap, \sqcup\}$ Voir le chapitre 4.	Le calcul de l'ensemble (fini) des meilleures couvertures est décidable (problème NP-Difficile).

$\mathcal{FLE}$ est la logique de description $\mathcal{FL}_0$ augmentée du constructeur de quantification existentielle qualifié $\exists R.C$.
$\mathcal{FL}^{\varepsilon}$ est une restriction du langage $\mathcal{FLE}$ définie pour éviter ses sources de complexité, voir la définition 32 de [39].
$\mathcal{AL}^{+}$ est défini comme $\mathcal{ALN}$ mais en remplaçant $\forall R.C$ par $\forall R.C \sqcap \geq 1\ R$.

TAB. 5.2 – Récapitulatif des réécritures existantes comparables avec les meilleures couvertures.



Les lettres en minuscules font référence aux articles suivants :
(a) → [44], (b) → [42], (c) → [56], (d) → [31], (e) → [69, 68], (f) → [12], (g) → [72], (h) → [39], (i) → [9], (j) → [41] et (k) → [21, 22, 23].

FIG. 5.1 – Les différentes réécritures existantes (la partie en grisé provient de [44]).

Une approche populaire pour résoudre la réécriture maximalement contenue, dans le cadre de la réécriture de requêtes en utilisant des vues, est l'approche par paniers issue du "bucket algorithm" étudié dans [55]. Le principe est repris et adapté ou amélioré dans [31, 72, 40, 69, 68, 39, 42][24]. Ce principe est de construire, pour chaque partie élémentaire du concept à récrire, un panier constitué des concepts définis qui récrivent cette partie. On réduit ainsi le nombre de réécritures candidates du concept entier en ne considérant que celles qui peuvent être construites avec un concept défini de chaque panier. Ainsi, il y a quatre étapes dans l'algorithme des paniers :

- on décompose le concept à récrire en parties élémentaires (cela correspond à la notion de clause pour les langages à subsomption structurelle et $\mathcal{ALN}$),
- à chaque partie élémentaires, on associe un paniers des concepts définis qui récrivent cette partie élémentaire,
- on fait le produit cartésien des paniers (phase très coûteuse) : chaque n-uplet obtenu constitue une réécriture candidate
- et on filtre l'ensemble des réécritures candidates pour ne garder que les réécritures maximalement contenues dans le concept à récrire (les réécritures candidates peuvent ne pas être subsumées par le concept à récrire, ou bien être inconsistantes).

L'application du l'algorithme des paniers est plus complexe dans [72, 40, 39, 42, 69, 68] dans le sens où, pour des raisons de complétude, les éléments des paniers ne sont plus des concepts définis (ou des vues) seuls, mais des ensembles de concepts définis ou de vues : ceci traduit le fait que l'obtention du panier de chaque partie élémentaire est déjà un problème de réécriture (à plus petite échelle).

L'étude des meilleures couvertures pour un langage ayant la propriété de subsomption structurelle suit globalement cet algorithme des paniers, en apportant une amélioration qui rend inutile l'étape de filtrage : le produit cartésien de l'algorithme des paniers suivi du filtrage des candidats obtenus selon un certain critère est remplacé par un calcul de transversaux minimaux de coût minimal. Ceci revient à fusionner le calcul du produit cartésien avec la vérification du critère de filtrage pour générer directement les meilleures réécritures (et uniquement celles-ci). A titre de comparaison, cette amélioration est du même type que celle induite par le théorème des persistants (voir l'algorithme 2 page 37) par rapport à la génération des transversaux minimaux d'un hypergraphe (voir l'algorithme 1 page 34).

L'approche par paniers telle qu'elle présentée précédemment n'est pas adaptée à la découverte des meilleures couvertures pour le langage $\mathcal{ALN}$. En effet, le filtrage a posteriori des réécritures inconsistantes pose problème. Supposons que l'on note n le nombre de parties élémentaires du concept à récrire. On construit donc n paniers. Que fait-on si tous les éléments du produit cartésien des paniers construits (non vides) correspondent à des couvertures inconsistantes (où l'inconsistance est découverte dans la phase de filtrage) ? Dans la réécriture maximalement contenue, aboutir à des réécritures toutes inconsistantes veut dire que l'instance étudiée du problème n'a pas de solution. Or on ne peut s'arrêter là dans la recherche des meilleures couvertures. En effet peut-être existe-t-il un ensemble de concepts définis élément du produit cartésien de m paniers parmi les n construits dont la conjonction n'est pas inconsistante. Cela signifie alors qu'il y a au moins une meilleure couverture. Mais trouver cette meilleure couverture, en appliquant l'algorithme des paniers implique, dans le pire des cas, de faire, pour tout m de n à 1 (en décrémentant de 1 à chaque fois), le produit cartésien de m paniers parmi n. Ceci implique au pire un nombre d'exécutions de l'algorithme des paniers de $2^n - 1$: si la construction des paniers est faite une fois pour toutes au début, le calcul des produits cartésiens (phase très

[24] L'algorithme "inverse rules" de [31] est basé sur un principe différent de celui de l'algorithme des paniers. Cependant, [44] montre qu'il est finalement proche de l'algorithme des paniers.

coûteuse), ainsi que le filtrage des réécritures non optimales ou inconsistantes sera effectué un nombre de fois exponentiel en fonction du nombre de parties élémentaires du concept à couvrir. Cela ne semble pas être vraiment envisageable. C'est pourquoi il faut essayer, en premier lieu, de restreindre l'espace de recherche à un espace où l'on est sûr que les éléments ne peuvent pas être inconsistants, et donc de trouver tous les cas d'inconsistance a priori.

Un cas nécessitant un nombre exponentiel d'exécutions de l'algorithme des paniers se présente quand, par exemple, les concepts définis de la terminologie sont deux à deux inconsistants. Ce cas illustre bien la nécessité de détecter tous les cas d'inconsistance avant d'effectuer le produit cartésien, c'est-à-dire avant de construire les réécritures candidates : les inconsistances entre couples de concepts définis vont impliquer les inconsistances entre m-uplets de concepts définis, pour $3 \leq m \leq n$. Connaître tous les cas d'inconsistances implique que l'on se limite à des réécritures composées d'un seul concept défini au maximum. Tester des réécritures de cardinalité 1 est alors bien plus rapide que de construire tous les m-uplets et de les tester.

Enfin, la troisième approche que nous évoquons ici est celle que nous proposons pour l'étude des meilleures couvertures avec le langage $\mathcal{ALN}$. Comme on l'a vu dans le chapitre 4, elle consiste à réduire par étapes successives l'espace de recherche pour finalement ne garder que les solutions. Les premières réductions sont issues de la découverte de tous les cas d'inconsistance, et les suivantes de la maximisation du rest. Dans les deux cas on construit des paniers et on effectue des calculs de transversaux minimaux afin de déterminer des ensembles de concepts qui définissent les contours de l'ensemble des meilleures couvertures dans l'espace de recherche.

L'algorithme MINICON, pour la réécriture de requêtes conjonctives en utilisant des vues [68], est basé sur une approche par paniers. Il est intéressant de l'évoquer ici car, à l'instar de *computeALNBCov*, il possède une étape de réduction a priori de l'espace de recherche. Grâce à cette réduction, MINICON améliore l'algorithme des paniers en assurant de bonnes performances lors du passage à l'échelle (quand le nombre de vues est important, jusqu'à plusieurs milliers). La réduction de l'espace de recherche consiste en un filtrage, avant le produit cartésien des paniers, de certains n-uplets qui aboutiraient à des réécritures non maximalement contenues. Ce filtrage s'effectue en examinant les jointures possibles entre les vues couvrant les parties élémentaires de la requête (appelées "subgoals"), et en assurant que les vues choisies pour récrire tous les subgoals de la requête forment une partition (et notamment ne s'intersectent pas). Les inconsistances éventuelles entre prédicats de comparaison sont aussi vérifiées avant combinaison des paniers pour éviter de générer des réécritures inconsistantes. Ainsi, le principe de réduire a priori l'espace de recherche avant de combiner les couvertures (vues ou concepts) des parties élémentaires de la requête (subgoals ou clauses) n'est pas nouveau. Dans l'algorithme MINICON, il permet de ne découvrir les vues inadéquates qu'une seule fois, comme dans *computeALNBCov* il permet de ne découvrir les inconsistances qu'une seule fois.

Cependant l'approche de filtrage utilisée dans l'algorithme MINICON n'est pas applicable à la recherche des meilleures couvertures, car elle est basée sur l'examen des variables apparaissant dans les subgoals et exprimant les jointures entre ces subgoals. Or, dans les descriptions de concept manipulées dans la découverte des meilleures couvertures, il n'y a ni variable ni jointure entre concepts. De plus, dans *computeALNBCov*, chaque clause peut être couverte au maximum par $p + 1$ S_i, où p est la profondeur maximale d'une clause, alors que dans MINICON, chaque subgoal ne peut être couvert que par une seule vue. D'une manière générale, on rappelle que les cas de meilleures couvertures sont plus variés que ceux des réécritures maximalement contenues, rendant les approches par paniers inadaptées (cf. le risque d'appliquer l'algorithme un nombre exponentiel de fois).

Le tableau 5.3 récapitule les approches algorithmiques évoquées et les travaux qui les utilisent.

	Nom de l'algorithme, ce qu'il calcule et référence	Principe de calcul	Traitement des inconsistances
Réé. équiv.	Un ensemble représentatif des réécritures contenant toutes les réécritures minimales. Voir [8, 9].	Extension puis réduction du concept à récrire.	Implicite à la méthode : étendre et réduire un concept consistant ne permet pas d'obtenir des réécritures inconsistantes.
Réécriture maximalement contenue	Un ensemble représentatif contenant au moins toutes les réécritures maximalement contenues. Voir [72, 40].	Algorithme des paniers adapté au contexte des requêtes conjonctives avec des prédicats qui sont des descriptions de concepts (notion de "descriptive support" pour passer d'un formalisme à l'autre).	Suppression des inconsistances a posteriori (pendant le filtrage).
	RRV (basé sur **RAV** et $\mathbf{RAV}_2$) et $\mathbf{RRV}^{\varepsilon}$ (basé sur $\mathbf{RAV}^{\varepsilon}$ et $\mathbf{RAV}_2^{\varepsilon}$) *RewriteAtom* et *RewriteQuery* Un ensemble représentatif contenant au moins toutes les réécritures maximalement contenues. Voir [39, 42].	Algorithme des paniers adapté au contexte des requêtes conjonctives avec des prédicats qui sont des descriptions de concepts (notion d'approximation faible ou de compilation pour passer d'un formalisme à l'autre).	Suppression des inconsistances après chaque produit cartésien (par ex. pendant la création des paniers et le filtrage).
	"Bucket algorithm" "Inverse rules algorithm" "MiniCon algorithm" Toutes les requêtes conjonctives maximalement contenues dans le concept à récrire et réunies en une disjonction forment la réécriture maximalement contenue (ou une réécriture maximalement contenue en présence de prédicat de comparaisons). Voir [56, 31, 69, 68, 44].	L'agorithme des paniers. L'algorithme des paniers amélioré dans le "MiniCon algorithm" par réduction de l'espace de recherche à la suite d'un examen des cas de jointures entre vues. Un algorithme proche des paniers avec l'"inverse rules algorithm".	Les inconsistances peuvent apparaître entre des prédicats de comparaison. Elles sont éliminées pendant la création des paniers dans l'algorithme MINICON.
Meilleures couvertures	*computeBCov* Exactement toutes les meilleures couvertures. Voir le chapitre 3.	Algorithme des paniers amélioré en un calcul de transversaux minimaux de coût minimal.	L'inconsistance est traitée comme une clause (i.e. non décomposable en une conjonction non triviale). Il n'y a donc aucun traitement particulier.
	computeALNBCov Exactement toutes les meilleures couvertures. Voir le chapitre 4.	Réductions successives de l'espace de recherche basées sur l'identification a priori des cas d'inconsistance.	La recherche des inconsistances se fait en début d'algorithme. Elle permet les premières réductions de l'espace de recherche (suppression des conjonctions inconsistantes).

TAB. 5.3 – Comparaison algorithmique des réécritures les plus proches des meilleures couvertures.

Chapitre 6

Application à la découverte dynamique de services web sémantique

Sommaire

Dans ce chapitre, nous présentons l'application des meilleures couvertures au problème de la découverte dynamique de services web sémantiques. La section 6.1 présente les notions de services web et de services web sémantiques, ainsi que le problème de la découverte dynamique de services web sémantiques. Dans la section 6.2, nous présentons les découvertes issues de l'application des meilleures couvertures au projet MKBEEM et à DAML-S.

6.1 Découverte dynamique de services web sémantiques

6.1.1 Service web

Apparus depuis quelques années à la suite d'initiatives à la fois industrielles et de recherche, les services web constituent un paradigme en pleine expansion pour l'intégration d'applications inter-entreprises au fil de l'eau sur internet. Dans ce paradigme, l'accent est mis sur le fait qu'elles possèdent elles-mêmes leur propre description, ce qui leur permet de se faire connaître, découvrir et utiliser facilement. Concrètement, un service web peut être défini comme une application rendue disponible sur internet par un fournisseur, et accessible à des clients [25]. Des services de réservation en ligne, de gestion de comptes bancaires, ou même des applications métiers entières en sont des exemples actuellement courants. L'ambition portée par les services web est de permettre une plus grande interopérabilité entre applications sur internet. On envisage ainsi

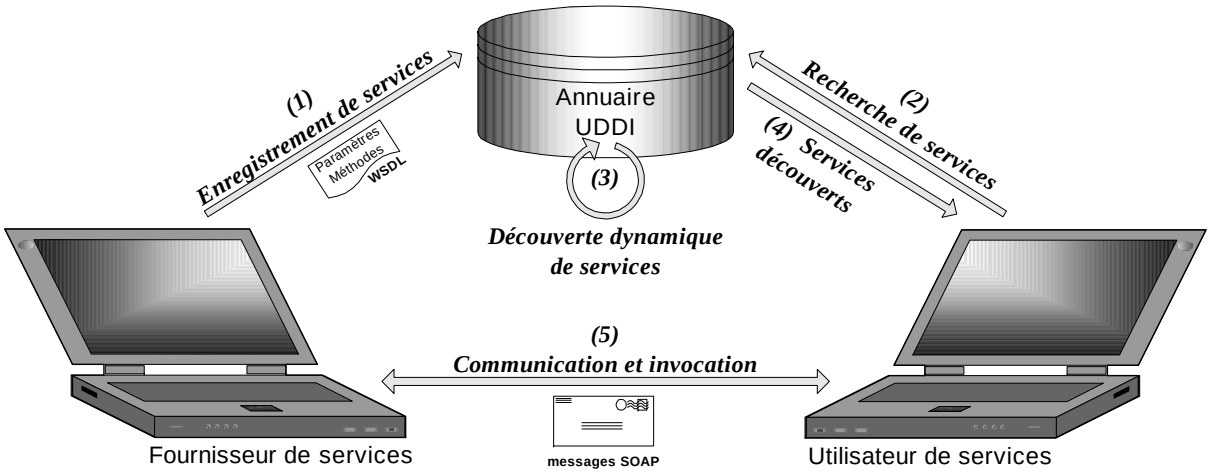


FIG. 6.1 – Fonctionnement des services web avec WSDL, UDDI et SOAP.

des services web capables, automatiquement, de se découvrir et d'être découverts, de négocier entre eux ou de se composer en des services plus complexes[24, 77].

Basée sur XML comme moyen d'échanger des messages entre applications, l'infrastructure actuelle des services web est constituée de trois principaux standards : SOAP (Simple Object Access Protocol)[25], WSDL (Web Services Description Language)[26] et UDDI (Universal Description, Discovery and Integration)[27]. SOAP est un protocole de transport de messages entre services web basé sur XML et http. WSDL est une syntaxe XML de description de la signature d'un service (ses entrées et ses sorties). UDDI est une norme définissant la description, la publication et la recherche de services web au sein d'annuaires (ou registres) dont la structure est aussi définie dans cette norme. La figure 6.1 montre le fonctionnement usuel de l'infrastructure des services web basée sur ces trois standard. Les fournisseurs enregistrent leurs services web dans un annuaire UDDI : chaque service est décrit par diverses informations dont la référence à un fichier WSDL qui va permettre aux clients intéressés de l'invoquer. L'utilisateur peut alors rechercher dans l'annuaire UDDI les services dont il a besoin par l'intermédiaire d'un mécanisme de recherche intégré dans l'annuaire (en général une recherche par mots-clés). Quand des services ont été découverts, l'utilisateur a accès à l'adresse et au fichier WSDL des services qui l'intéressent, et il peut alors entrer en contact direct avec le fournisseur et invoquer directement le service. La communication entre son application et le service est alors assurée par des messages SOAP.

6.1.2 Service web sémantiques

L'infrastructure des services web présentée précédemment facilite l'implémentation, la publication et l'invocation de services. Cependant, elle ne permet pas encore aux services web de tenir leur promesse d'un fonctionnement largement automatisé. Cette automatisation est pourtant essentielle pour faire face aux exigences du passage à l'échelle, d'une forte réactivité dans un environnement hautement dynamique et de la volonté de réduire les coûts de développement et de maintenance des services.

[25]Voir http ://www.w3.org/TR/soap12-part0/.
[26]Voir http ://www.w3.org/TR/wsdl/.
[27]Voir http ://www.uddi.org/.

Dans le cadre du web sémantique [17] dont le but est de baser les traitements d'informations du web sur la définition de leur sémantique et sur les raisonnements qui en découlent, de nouveaux langages pour une description formelle des services web ont été proposés. Les logiques de description y occupent une place centrale grâce aux différents niveaux d'expressivité qu'elles permettent et au nombre important de raisonnements dont elles disposent. Ces raisonnements sont autant d'outils pouvant permettre d'automatiser certaines tâches dans le fonctionnement des services web. C'est ainsi qu'est née la notion de service web sémantique.

Un service web sémantique est donc décrit par un langage qui a une sémantique formelle, ce qui implique l'absence d'ambiguïté dans ce qui est décrit et ainsi la possibilité de raisonner par rapport à ces descriptions. En général, la description d'un service web sémantique est construite avec des termes qui ont eux-mêmes une description formelle et qui sont regroupés dans des dictionnaires structurés appelés ontologies. La sémantique d'un service web est alors le sens de sa description par rapport à la sémantique du langage utilisé ainsi que par rapport à la sémantique des termes employés dans sa description.

Il existe plusieurs langages logiques pour décrire des services web. Un des plus utilisés est le langage DAML+OIL [45, 46]. DAML+OIL est basé sur une logique de description très expressive (appelée $\mathcal{SHIQ}$). DAML+OIL a servi dans de nombreux travaux concernant la représentation et la découverte de services web sémantiques, directement ou au travers de DAML-S. DAML-S est une ontologie de concepts DAML+OIL décrivant les aspects techniques d'un service web (entrées/sorties, paramètres, types de données,...)[1]. Le langage OWL, évolution directe de DAML+OIL, a récemment été normalisé par le W3C : c'est désormais le principal langage standard de description d'ontologies sur le web[28]. OWL-S est l'évolution correspondante de DAML-S. D'autres approches logiques, comme [60] pour les logiques de description ou [26] pour Prolog, viennent encore confirmer l'intérêt des logiques dans ce domaine.

Dans l'utilisation de ces langages, on peut distinguer trois approches complémentaires de description des services web sémantiques, qui se distinguent selon leur caractère abstrait/concret et implicite/explicite (la distinction implicite/explicite vient de [73]) :

- Description abstraite des services web : on décrit un service par rapport aux concepts du domaine d'application, eux-mêmes définis dans l'ontologie du domaine, et par rapport aux contraintes du service, comme on pourrait le faire en langage naturel. Par exemple, à la manière de [75], la compagnie $ExCie$ qui décrit, pour publication dans un registre, un service de vente et de livraison d'un ordinateur, pourrait le faire de la manière suivante :

$$\begin{aligned} Vente\_Livraison\_PC \equiv\ & \exists Publication \sqcap aService.(Vente \sqcap \exists vendeur.ExCie \sqcap \\ & \exists objet.(PC \sqcap \exists memoire.PlusDe128Mo) \sqcap \\ & \exists prix.EnDessous700 \sqcap \exists quantite.Endessous200 \sqcap \\ & \exists estComposeDe.(Livraison \sqcap \exists date.Avant31oct2004 \sqcap \\ & \exists lieu.Clermont)) \end{aligned}$$

  où les concepts $Publication$, $Vente$, $Livraison$, ..., sont définis dans l'ontologie du domaine.
- Description concrète implicite des services : la description des services est celle de leurs fonctionnalités (en anglais "capability representation"), où ces fonctionnalités sont décrites par les transformations qu'elles induisent, c'est-à-dire en termes d'états initiaux/finaux, d'entrées/sorties et de préconditions/postconditions. Par exemple, dans [66], un service (simple) de vente de voiture est décrit par une entrée qui est le prix et par une sortie qui est la voiture vendue, où prix et voiture sont des concepts définis dans l'ontologie du

[28] Voir http ://www.w3.org/TR/owl-ref/.

domaine, et où les notions d'entrée et de sortie sont définies dans une ontologie concernant les aspects techniques des services.

– Description concrète explicite des services : la description des services est celle de leurs fonctionnalités décrites explicitement au sein d'un modèle de processus par les noms des tâches, ou processus, et sous-tâches qu'ils assurent (i.e. à partir desquels ils sont construits). Dans ce cas, on a besoin non d'une ontologie du domaine, mais d'une ontologie de processus décrivant chaque processus qui compose la description du service. C'est typiquement le cas de [18, 48] qui s'appuient sur l'ontologie de processus du MIT Process Handbook[29] pour décrire les services à partir de processus extraits de cette ontologie. Par exemple, un service de vente sur internet sera représenté par les sous-tâches d'identification de clients (via des techniques de fouille de données), d'entrée en contact avec les clients et d'enregistrement des commandes (chaque sous-tâche pouvant être à son tour décrite en termes de resources utilisées, d'exceptions,...). Les trois sous-tâches précédentes sont définies dans l'ontologie des processus.

On résume au travers du tableau 6.1, les caractéristiques des trois approches complémentaires de description des services web sémantiques, en positionnant les articles précédemment cités.

6.1.3 Découverte dynamique

Le problème de la découverte dynamique des services web est central dans le fonctionnement des services web. En supposant que l'on a un certain nombre de descriptions de services web provenant de fournisseurs divers, ainsi qu'une requête d'un utilisateur, comment découvrir parmi tous les services, ceux qui correspondent à la requête de l'utilisateur ? On qualifie cette découverte de dynamique puisque l'utilisateur ne sait pas a priori quels services il désire obtenir et puisque l'offre de services peut changer très rapidement, le contexte du web étant très versatile. Ainsi, pour une même requête, à des instants différents, les services découverts ne seront pas toujours les mêmes, puisque ceux-ci peuvent à tout moment être modifiés, supprimés ou ajoutés.

Les méthodes classiques de découverte dynamique de services sont syntaxiques, c'est-à-dire basées sur la recherche et la comparaison de chaînes de caractères. [18, 48] donnent un panorama intéressant de ces techniques. Elles sont de deux types : les recherches par mots-clés, très utilisées par les moteurs de recherche, et les recherches pas tables de couples (clé,valeur), utilisées par exemple dans UDDI, où une correspondance est établie si deux clés ont la même valeur pour deux services différents. Dans [18, 48], ces approches sont étudiées par rapport aux critères de précision et de complétude de l'ensemble des correspondances trouvées. Sa conclusion résulte en une bonne complétude de l'ensemble des réponses pour une précision très mauvaise.

La description des services avec des langages logiques permettant de raisonner a permis d'améliorer la pertinence des découvertes. Ainsi, les services web sémantiques permettent la définition de critères sémantiques de découverte. Par exemple, on peut imaginer vouloir découvrir des services qui ne sont pas incohérents (au sens logique) avec la requête. Il ne s'agit donc plus d'une simple comparaison de chaînes de caractères.

L'usage de langages logiques très expressifs pour décrire les services permet d'imaginer nombre de critères sémantiques de découverte. Voici les qualités importantes à atteindre qui doivent guider la définition de ces critères de découverte [73] :

– flexibilité : la découverte doit être la plus flexible possible,

[29] L'ontologie de processus du MIT Process Handbook project est en fait une taxinomie des processus organisationnels de certains domaines où les processus sont reliés par des relations de spécialisation/généralisation et de composition. Voir http ://ccs.mit.edu/ph/.

	Approche abstraite	Approche concrète implicite	Approche concrète explicite
Avantages	• proche du langage naturel : – plus intuitif pour un utilisateur – couplage avec un module de traitement de la langue naturelle • autorise des critères de découverte fins	• souplesse de modélisation – ajout facile de nouvelles contraintes – s'adapte à tous les domaines • ontologies de tailles moyennes (décrivant un domaine de façon abstraite et non par tous les processus possibles qu'on peut y trouver) • proche de l'implémentation • autorise des critères de découverte fins	• réduit la difficulté de modélisation (choix des tâches dans l'ontologie de processus) • plus l'ontologie est grande et détaillée, plus la découverte va être facilitée (recherche de correspondances en termes de l'ontologie dans la requête et les services) • intéressant dans des domaines précis où les processus sont bien connus
Inconvénients	• mélange d'informations de natures différentes (concepts, valeurs, entrées/sorties,...)	• découverte plus complexe qu'avec une ontologie de processus • description assez technique du service, pas toujours intuitive	• critères de découverte simplifiés (essentiellement mise en correspondance des processus décrivant la requête et les services) • ontologies de grandes tailles difficiles à manipuler • difficulté d'ajouter des contraintes aux processus • nécessite une ontologie par domaine
Types d'ontologies utilisées	Ontologie abstraite du domaine (définition des concepts et notions importantes du domaine) Par exemple, une ontologie concernant le matériel informatique contiendra les définitions des concepts ordinateur, PC, mémoire,... Une ontologie du domaine de la vente contiendra les définitions de prix, vendeur, acheteur, délai,...	Ontologie concrète du domaine définissant les aspects plus techniques du service, comme les notions d'entrées/sorties, de paramètres, de types de données,... Exemple : DAML-S [1]. Une ontologie du domaine abstraite est en général utilisée conjointement. Par exemple, quand on décrit une sortie qui est une voiture, l'ontologie concrète (par exemple DAML-S) définit sortie, et l'ontologie abstraite (par exemple du domaine des transports) définit voiture.	Ontologie de processus où sont décrits tous les processus qu'on peut trouver dans un domaine particulier (comme les finances, la gestion de ressources humaines,...). Exemples : le MIT Process Handbook, RosettaNet (voir http ://www.rosettanet.org/). Ce sont en fait des taxinomies de processus où les concepts sont classés par spécialisation/généralisation, et non pas définis dans un langage possédant une sémantique. Pour les aspects techniques, une ontologie est parfois utilisée pour décrire les services plus précisément qu'une liste de processus.
Références et langages	• [43] utilise DAML+OIL. • [26] utilise DAML + Prolog. • [29] utilise DAML-S. • [75] utilise DAML+OIL. • [58] utilise DAML+OIL et DAML-S (la description abstraite des services définit les sorties du service). • Notre approche (voir la section 6.2.1) utilise les logiques de description ayant la propriété de subsomption structurelle.	• [58] utilise DAML+OIL et DAML-S (les entrées décrivent les paramètres d'exécution du service). • [1, 67, 66, 73] définissent et utilisent DAML-S. • [11] utilise DAML-S. • Notre approche (voir la section 6.2.2) utilise les sous-langages de DAML-S qui ont la propriété de subsomption structurelle.	• [18, 48] utilisent l'ontologie du MIT Process Handbook pour indexer et faire référence aux processus qu'ils définissent, une petite ontologie définissant les aspects techniques des services, ainsi qu'un langage d'interrogation de processus nommé PQL (pour "Process Query Language") qui est très proche de la logique du premier ordre limitée aux prédicats unaires et binaires (et donc des logiques de description).

Tab. 6.1 – Approches de description de services web sémantiques.

– précision et complétude : elle doit éviter l'obtention de services non désirés (en anglais les "false positives") et la non obtention de services désirés (en anglais les "false negatives"), et ce en dépit de sa flexibilité et
– efficacité : elle doit être implémentée par des mécanismes efficaces pour ne pas ennuyer l'utilisateur avec des délais de recherche trop longs.

Un certain nombre d'approches dans la description et la découverte sémantique de services web ont été récemment suggérées. On citera notamment [26, 43, 29, 75, 18, 48, 11, 66, 73, 58]. Globalement, deux idées ressortent de ces approches :

– les services découverts doivent en général être cohérents avec la requête (i.e. il ne doit pas y avoir d'inconsistance logique entre les services et la requête) et
– la qualité des services découverts se juge par rapport aux éventuelles relations de spécialisation/généralisation (i.e. la subsomption pour les logiques de descriptions) existant entre les services et la requête.

A la section suivante, nous présentons les applications des meilleures couvertures à la découverte sémantique et nous les positionnons par rapport aux travaux les plus proches.

6.2 Applications des meilleures couvertures

Nous proposons deux applications de la découverte des meilleures couvertures pour résoudre le problème de la découverte dynamique de services web sémantiques. La première application a été réalisée dans le contexte du projet européen MKBEEM [53, 28] et se positionne dans l'approche abstraite pour la description des services web. Elle concerne les logiques de description ayant la propriété de subsomption structurelle et est présentée dans la section 6.2.1. La seconde est l'adaptation des meilleures couvertures à DAML-S, et plus particulièrement aux services décrits en termes d'entrées et de sorties définies dans la partie profile de DAML-S : c'est la découverte des meilleures couvertures de profile. Elle suit l'approche concrète implicite de description des services. Elle est présentée dans la section 6.2.2.

Notons que nous avons aussi appliqué les meilleures couvertures à la recherche de catalogues électroniques (ou "e-catalog" en anglais) dans un contexte pair-à-pair [13, 10, 14]. Nous avons montré que l'exploitation des rest (i.e. les parties non résolues des requêtes) était tout-à-fait adaptée à ce contexte dans le sens où l'infrastructure pair-à-pair facilitait grandement la diffusion de ces rest à tous les pairs, et où la sémantique des liens entre pairs permettait de guider les rest vers des pairs susceptibles de pouvoir leur fournir des réponses.

6.2.1 Les meilleures couvertures dans MKBEEM

Après la présentation du projet MKBEEM, nous explicitons au travers d'un exemple comment les meilleures couvertures sont utilisées pour la découverte de services et nous comparons l'approche avec les travaux existants.

Le projet MKBEEM

Le projet européen MKBEEM[30] [53, 28] a pour but l'étude et la réalisation de plateforme de commerce électronique, basée sur des techniques de traitement de la langue naturelle et de représentation de connaissances et de raisonnement. Ces plateformes offrent des services de médiation multilingues et intelligents dans le domaine du tourisme et de la vente par correspondance. Dans

[30] MKBEEM est l'acronyme de Multilingual Knowledge Based European Electronic Marketplace (IST-1999-10589, 1er Fév. 2000 - 1er Déc. 2002). Voir http ://www.mkbeem.com

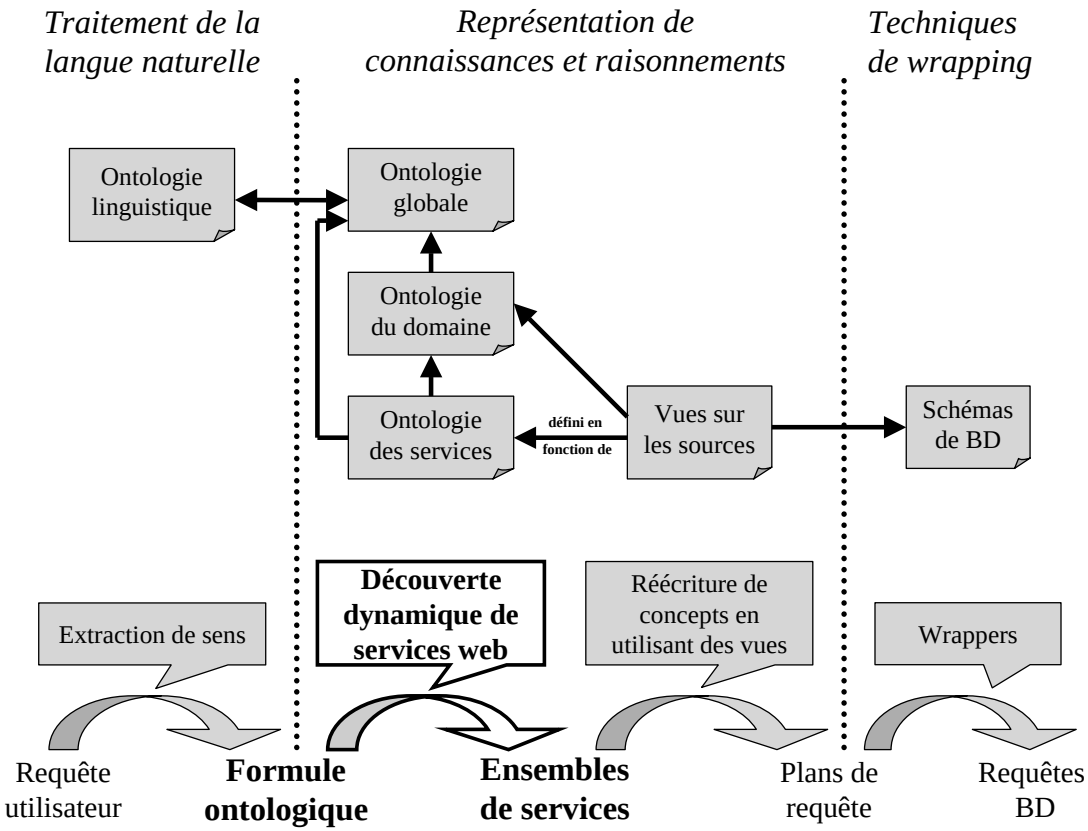


FIG. 6.2 – Les étapes de la médiation dans MKBEEM.

cette médiation, le problème de découverte dynamique de services est lié à d'autres problèmes comme l'interprétation du sens du langage naturel et la génération de plans de requêtes. Plus précisément, les étapes de médiation dans MKBEEM sont les suivantes :

- Un utilisateur exprime une requête en langue naturelle.
- Par des techniques de traitement de la langue naturelle, elle est transformée en une formule ontologique (qui est en réalité une formule de logique du premier ordre avec des prédicats unaires et binaires) : c'est l'étape d'extraction du sens, grâce à une ontologie linguistique.
- Cette formule ontologique est alors transformée en une description de concept Q équivalente (ou la plus proche possible) de $\mathcal{FL}_0$ (qui a la propriété de subsomption structurelle). Grâce à une ontologie $\mathcal{T}$ de services exprimés aussi avec le langage $\mathcal{FL}_0$ (c'est donc une $\mathcal{FL}_0$-terminologie), on peut rechercher les meilleures couvertures de Q en utilisant $\mathcal{T}$. On obtient alors toutes les combinaisons de services les plus proches de Q. C'est ici que le miss est utilisé pour établir un dialogue avec l'utilisateur. On rappelle que le miss contient les parties des descriptions de services qui ne correspondent à aucune parties de la requête. L'utilisateur doit donc renseigner ces parties, par exemple grâce à des formulaires créés dynamiquement, pour un bon fonctionnement ultérieur des services.
- Pour chaque combinaison, on découvre des plans de requêtes à partir de la définition de vues décrivant des sources et reliées dans l'ontologie aux descriptions des services. C'est le système PICSEL qui est ici utilisé [40], et qui effectue cette tâche par une technique de

recherche des réécritures de requêtes maximalement contenues en utilisant des vues[31].
- Enfin, à partir des plans de requêtes et de wrappers ad-hoc, on peut exécuter les requêtes sur les systèmes liés au médiateur et obtenir les résultats.

Les connaissances manipulées durant la découverte de services sont structurées en une ontologie formée de trois parties : l'ontologie globale (contenant les définitions des concepts génériques), l'ontologie du domaine (celle des concepts du tourisme ou de la vente par correspondance) et l'ontologie des services (voir [28] pour plus de détails). La figure 6.2 récapitule le déroulement de cette médiation et les connaissances utilisées à chaque étape.

Dans les premières expériences menées avec le prototype MKBEEM, nous avons modélisé de petites ontologies (environ 500 concepts définis et 50 services web) sur le domaine du tourisme notamment. Le traitement des requêtes utilisateurs s'est avéré qualitativement satisfaisant, les combinaisons de services web découverts étant pertinentes. De plus, le temps total dédié à la découverte n'a jamais dépassé quelques secondes (pour des résultats plus détaillés, se référer au chapitre 7 page 117).

Voyons maintenant un exemple précis.

Exemple

L'exemple est situé dans le domaine du tourisme. Le tableau 6.2 nous donne la définition dans la logique de description $\mathcal{FL}_0$ d'une ontologie $\mathcal{T}$ contenant des définitions de concepts et de services relatifs au tourisme ainsi qu'une requête Q possible.

A partir de l'ontologie et de la requête données au tableau 6.2, le processus de découverte se poursuit en 4 étapes. L'étape 1 est le dépliage de l'ontologie et de la requête, ainsi que la normalisation (i.e. mise sous forme RCF) de toutes les descriptions (pour $\mathcal{FL}_0$ c'est l'application exhaustive de la règle $\forall R.(A \sqcap B) \rightarrow \forall R.A \sqcap \forall R.B$). Le tableau 6.3 montre le résultat de l'étape 1 pour la requête. L'étape 2 consiste en la construction de l'hypergraphe $\mathcal{H}_{\mathcal{T}Q}$ à partir des clauses de la requête. On rappelle que les sommets de cet hypergraphe sont les services web, les arêtes e_i sont les clauses de la requête, et un sommet est dans une arête si le service correspondant contient la clause correspondante dans sa RCF. La figure 6.3 donne l'hypergraphe correspondant. Les transversaux minimaux de cet hypergraphe sont ensuite calculés durant l'étape 3, afin de trouver les combinaisons de services qui maximisent l'information commune avec la requête (i.e. celles qui minimisent la taille du rest). Dans l'exemple courant, les transversaux minimaux de cet hypergraphe sont, d'après la figure 6.3, les combinaisons 1 et 2, soit respectivement $\{Horaire\_Arrivee, Hotel\}$ et $\{Horaire\_Arrivee, Appartement\}$. Enfin, l'étape 4 consiste en l'identification parmi ces combinaisons de celles qui minimisent l'information apportée en plus par rapport à la requête (i.e. le miss). Dans notre exemple, $\{Horaire\_Arrivee, Appartement\}$ est ainsi plus proche de la requête que $\{Horaire\_Arrivee, Hotel\}$ puisqu'$Hotel$ contient une clause en plus qui est $\forall equip\_loisirs.Television$. La meilleure combinaison de services pour répondre à la requête est donc $\{Appartement, Horaire\_Arrivee\}$.

Comme nous pouvons le voir sur le tableau 6.4, l'utilisateur peut connaître le rest de la requête, i.e. la partie de sa requête qui n'a pas été comprise, et le miss, i.e. la partie de la combinaison des services web qui n'a pu être mise en correspondance avec aucun élément de la requête. Le miss sert à initier un dialogue avec l'utilisateur afin que ce dernier donne d'éventuels renseignements nécessaires au fonctionnement du service et qu'il n'aurait pas précisés dans sa requête initiale. Le rest peut servir à améliorer les services découverts si on en fait une requête pour une nouvelle découverte dynamique sur une autre ontologie de services. On peut alors

[31] On rappelle qu'à l'instar de la découverte des meilleures couvertures, la réécriture implémentée dans PICSEL est aussi une instance du cadre formel de la réécriture dans les logiques de description [9].

Ontologie générale

$Heure$	$\equiv$	$\forall heures.Entier \sqcap \forall minutes.Entier$
$Date$	$\equiv$	$\forall num\_jour.Entier \sqcap \forall num\_mois.Entier \sqcap \forall num\_annee.Entier \sqcap \forall jour\_sem.Chaine\_car \sqcap \forall nom\_mois.Chaine\_car$

Ontologie du domaine du tourisme

$Voyage$	$\equiv$	$\forall lieu\_dep.Chaine\_car \sqcap \forall lieu\_arr.Chaine\_car \sqcap \forall moyen\_transport.Chaine\_car$
$Logement$	$\equiv$	$\forall lieu\_sejour.Chaine\_car \sqcap \forall premier\_jour.Date$

Ontologie de services web

$Hotel$	$\equiv$	$Logement \sqcap \forall nb\_lits.Entier \sqcap \forall categorie.Chaine\_car \sqcap \forall equip\_loisirs.Television$	Permet de consulter une liste d'hôtels dont les chambres sont équipées d'une télévision, en donnant quelques informations comme le lieu, la date du premier jour, le nombre de lits ou la catégorie de l'hôtel.
$Appartement$	$\equiv$	$Logement \sqcap \forall nb\_chambres.Entier \sqcap \forall categorie.Chaine\_car$	Permet de consulter une liste d'appartements en donnant quelques informations comme le lieu, la date du premier jour, le nombre de chambres et la catégorie d'appartement.
$Horaire\_Depart$	$\equiv$	$Voyage \sqcap \forall heure\_dep.Heure \sqcap \forall date\_dep.Date$	Permet de consulter une liste de voyage étant donné les lieux de départ et arrivée et l'heure et la date de départ.
$Horaire\_Arrivee$	$\equiv$	$Voyage \sqcap \forall heure\_arr.Heure \sqcap \forall date\_arr.Date$	Permet de consulter une liste de voyage étant donné les lieux de départ et arrivée et l'heure et la date d'arrivée.

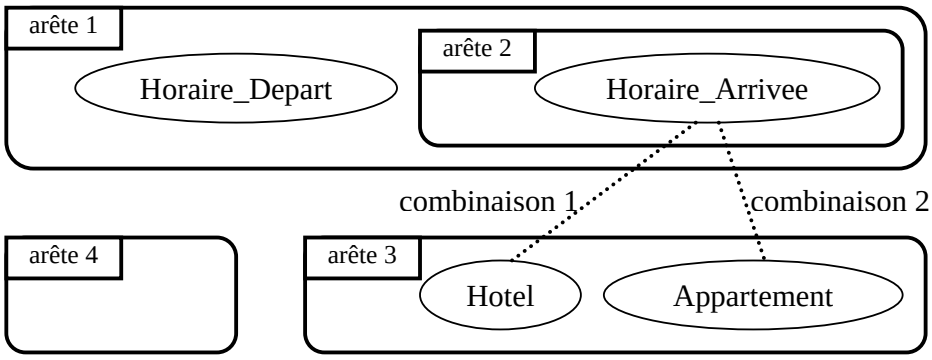
Une requête possible

"J'arriverai à Paris le lundi 29 octobre et je voudrais un logement avec piscine." Demande de services web contenant un lieu d'arrivée décrit par une chaîne de caractères, associé avec une date d'arrivée décrite par un nom de jour, un numéro de jour et un nom de mois, et un logement associé à un équipement de loisir qui est une piscine.	$Requete\ Q$	$\equiv \forall lieu\_arr.Chaine\_car \sqcap \forall date\_arr.Requete\_partie\_1 \sqcap Logement \sqcap \forall equip\_loisirs.Piscine$
	$Requete\_partie\_1$	$\equiv \forall num\_jour.Entier \sqcap \forall jour\_sem.Chaine\_car \sqcap \forall nom\_mois.Chaine\_car$

TAB. 6.2 – Une petite ontologie $\mathcal{T}$ du tourisme et une requête Q.

Requete Q	$\equiv$	$\forall lieu\_arr.Chaine\_car \sqcap$ $\forall date\_arr.\forall num\_jour.Entier \sqcap$ $\forall date\_arr.\forall jour\_sem.Chaine\_car \sqcap$ $\forall date\_arr.\forall nom\_mois.Chaine\_car \sqcap$ $\forall lieu\_sejour.Chaine\_car \sqcap$ $\forall premier\_jour.\forall num\_jour.Entier \sqcap$ $\forall premier\_jour.\forall num\_mois.Entier \sqcap$ $\forall premier\_jour.\forall num\_annee.Entier \sqcap$ $\forall premier\_jour.\forall jour\_sem.Chaine\_car \sqcap$ $\forall premier\_jour.\forall nom\_mois.Chaine\_car \sqcap$ $\forall equip\_loisirs.Piscine$

TAB. 6.3 – La requête normalisée (après dépliage et normalisation de sa description).



L'arête 1 correspond à la clause $\forall lieu\_arr.Chaine\_car$

L'arête 2 correspond aux clauses $\forall date\_arr \forall jour\_sem.Chaine\_car$
$\forall date\_arr \forall nom\_mois.Chaine\_car$
$\forall date\_arr \forall num\_jour.Entier$

L'arête 3 correspond aux clauses $\forall lieu\_sejour.Chaine\_car$
$\forall premier\_jour \forall jour\_sem.Chaine\_car$
$\forall premier\_jour \forall nom\_mois.Chaine\_car$
$\forall premier\_jour \forall num\_mois.Entier$
$\forall premier\_jour \forall num\_jour.Entier$
$\forall premier\_jour \forall num\_annee.Entier$

L'arête 4 correspond à la clause $\forall equip\_loisirs.Piscine$

FIG. 6.3 – Hypergraphe $\mathcal{H}_{\mathcal{T}Q}$ construit à partir de l'ontologie $\mathcal{T}$ et de la requête Q du tableau 6.2, et les transversaux minimaux associés (combinaisons 1 et 2).

Meilleure combinaison de services	**Rest : partie de la requête absente de tout service**	**Miss : partie de la combinaison absente de la requête**
$\{Horaire\_Arrivee$, $Appartement\}$	$\forall equip\_loisirs.Piscine$	$\forall categorie.Chaine\_car$ $\forall lieu\_dep.Chaine\_car$ $\forall moyen\_transport.Chaine\_car$ $\forall heure\_arr.\forall heures.Entier$ $\forall heure\_arr.\forall minutes.Entier$ $\forall date\_arr.\forall num\_mois.Entier$ $\forall date\_arr.\forall num\_annee.Entier$ $\forall nb\_chambres.Entier$

Tab. 6.4 – Résultats de la découverte dynamique de services web.

obtenir d'autres services à combiner avec les premiers pour avoir des combinaisons couvrant entièrement la requête initiale.

A partir de l'exemple qui vient d'être exposé, nous pouvons faire deux remarques sur le fonctionnement de *computeBCov*. Premièrement, on peut avoir l'impression qu'une partie significative de la requête, quand celle-ci est exprimée en langage naturel, est perdue, ou en tout cas non prise en compte dans la recherche de services. En effet, les expressions "Paris" et "lundi 29 octobre" ne figurent pas dans la requête exprimée comme une $\mathcal{FL}_0$-description. En réalité, ces parties y figurent indirectement avec les concepts $Chaine\_car$ et $Requete\_partie\_1$. Avant d'exécuter *computeBCov*, le prototype MKBEEM extrait les valeurs de la requête ("Paris", "lundi", "29" et "octobre") pour que cette dernière soit plus conceptuelle et corresponde ainsi à la manière avec laquelle les services ont été décrits. Dans le contexte de MKBEEM, l'expérience a en effet montré que les valeurs n'étaient pas utiles pour décrire des services, et qu'il était plus pertinent de découvrir des services en sachant que l'utilisateur avait précisé une date, plutôt qu'en connaissant précisément cette date. D'une manière générale, le problème qui se pose est celui du niveau de granularité de la modélisation. Si l'on suppose que l'on utilise une ontologie du tourisme de grande taille et très détaillée, alors on peut supposer, par exemple, que "Paris" a une définition dans cette ontologie, et que certains services sont définis avec ce concept. Il faut alors utiliser ce concept dans l'expression de la requête (i.e. ne pas considérer "Paris" comme une valeur). Notons toutefois qu'après découverte des meilleures couvertures, les valeurs mises de côté sont réassociées avec leur concepts d'origine et réutilisées par le système PICSEL, conjointement avec les ensembles de services découverts, pour calculer des plans de requêtes à partir de vues des bases de données disponibles (voir la figure 6.2 page 107).

La deuxième remarque concerne le résultat obtenu, et plus particulièrement le fait d'obtenir *Appartement* préférentiellement à $Hotel$. La description d'$Hotel$ précise un équipement de loisir (une télévision) alors qu'*Appartement* n'en a aucun. Comme la requête en demande un (une piscine), même si ce n'est pas le même, il serait intuitivement plus intéressant de proposer $Hotel$ plutôt qu'*Appartement*. Or *computeBCov* ne découvre pas $Hotel$ car les mises en correspondance qu'il effectue sont basées sur les clauses. Ainsi, même si un rôle d'une clause est commun à un rôle d'une autre clause, les deux clauses sont considérées entièrement distinctes. Ceci provient de l'expressivité du langage utilisé et de la définition de la différence. Ici le langage $\mathcal{FL}_0$ n'est pas assez expressif pour que la différence prenne en compte la notion commune d'équipement de loisir. Si on avait pu utiliser le constructeur $(\geq nR)$, on aurait pu ajouter la clause $\geq 1 equip\_loisirs$ à

Hotel et à la requête, et on aurait alors découvert *Hotel* au lieu de *Appartement*.

Avant d'évoquer les autres approches de découvertes de services, nous renvoyons à l'annexe E page 195 pour des scenarii supplémentaires de découverte de services dans le contexte de MKBEEM. Dans cette annexe, nous donnons une ontologie du tourisme contenant 27 services et 25 concepts définis, modélisés sur le modèle de ceux de l'ontologie de MKBEEM. Nous étudions quatre cas de requêtes qui illustrent les principales caractéristiques de *computeBCov* (flexibilité, inférence, importance des rest et miss, et combinatoire). Ces exemples sont expliqués à l'aide du prototype BCover présenté à la section 7.1 page 117.

Positionnement par rapport aux autres approches

Nous comparons cette application des meilleures couvertures avec les trois plus proches travaux parmi ceux évoqués précédemment qui sont [43, 75, 58]. La comparaison est détaillée au tableau 6.5. Les trois approches décrivent aussi des decouvertes de services web. Elles sont comparées sur la base des critères suivants :

- La correspondance avec Q : pour une requête Q, on découvre soit plusieurs services "séparés", c'est-à-dire que chacun individuellement répond à Q, soit des combinaisons de services, c'est-à-dire que dans chaque combinaison, c'est l'ensemble des services qui répond à Q.
- La définition (de la correspondance avec Q) : c'est la condition à satisfaire pour qu'un service ou une combinaison de service soit considérée comme une réponse possible à Q.
- Le langage : c'est le langage utilisé pour décrire Q et les services.
- Le degré de correspondance : ce sont une ou plusieurs relations d'ordre qui sont définir pour classer les solutions les unes par rapport aux autres.
- Les propriétés intéressantes : ce sont des potentialités caractéristiques de certaines approches.

Les principales conclusion que l'on peut tirer de cette comparaison avec les travaux existants sont les suivantes. La découverte des meilleures couvertures est l'approche :

- la plus générale et flexible en ce qui concerne le fonctionnement de la découverte : c'est la seule à découvrir des combinaisons de services, et sa définition de la correspondance entre Q et les solutions est très souple (puisqu'une combinaison de services n'ayant aucun lien de subsomption avec la requête peut quand même être solution, et puisque les rest et miss sont des descriptions réutilisables).
- la plus restreinte en ce qui concerne l'expressivité des langages utilisés (les logiques de description ayant la propriété de subsomption structurelle).

6.2.2 Les meilleures couvertures dans DAML-S

Dans cette section, nous présentons l'application des meilleures couvertures à des services décrits avec DAML-S. DAML-S[32] [1] est une ontologie de concepts DAML+OIL permettant de décrire le profile d'un service (principalement ses entrées et sorties), le modèle d'un service (la description du service en tant que processus en termes d'enchaînement de tâches à exécuter), et les fondements du service (comment communiquer avec lui). Dans [15], nous avons montré comment la découverte des meilleures couvertures peut s'adapter à la description d'un service selon les entrées et sorties du profile DAML-S [13, 10, 14]. Le principe, illustré à la figure 6.4 est le suivant :

[32]DAML+OIL ayant récemment été renommé OWL dans le cadre d'une recommandation W3C (voir http ://www.w3.org/TR/owl-features/), DAML-S a de même été renommé OWL-S.

Approche	• Correspondance avec Q • Définition • Langage	Degré de correspondance	Propriété intéressante
[43]	• des services S séparés • $Q \sqcap S$ satisfiable • DAML+OIL	Du meilleur au pire : – exact : $Q \equiv S$ – subsume : $Q \sqsupseteq S$ – plug-in : $Q \sqsubseteq S$ – S sous-consept d'un super-concept de Q – disjoint : $Q \sqcap S \sqsubseteq \bot \Rightarrow S$ inacceptable Appliqué aux descriptions de services DAML+OIL.	
[75]	• des services S séparés • $Q \sqcap S$ satisfiable • DAML+OIL	Pas de classement. Sur la base des informations en commun T entre Q et S ($T \equiv Q \sqcap S$) on reformule Q en une proposition P devant suivre $P \sqsubseteq T$.	Cadre général de la découverte et de la négociation.
[58]	• des services S séparés • $Q \sqcap S$ satisfiable • DAML+OIL et DAML-S	Du meilleur au pire : – exact : $Q \equiv S$ – plug-in : $Q \sqsubseteq S$ – subsume : $Q \sqsupseteq S$ – intersection : $\neg(Q \sqcap S) \sqsubseteq \bot$ – disjoint : $Q \sqcap S \sqsubseteq \bot \Rightarrow S$ inacceptable Appliqué aux sorties du profile DAML-S.	
Meill. couvertures	• des combinaisons E de services • Au moins une information en commun entre Q et E : $Q - lcs(Q, E) \not\equiv Q$ • Toutes les LD à subsomption structurelle	Minimisation successives de : 1) la taille de $Rest_E(Q)$: i.e. du meilleur au pire : – exact : $Q \equiv E$ – subsume : $Q \sqsupseteq E$ – plug-in : $Q \sqsubseteq E$ – disjoint : $Q \sqcap E \sqsubseteq \bot \Rightarrow E$ inacceptable – si pas de disjonction, E peut être acceptée si pas de E' meilleur. 2) la taille de $Miss_E(Q)$	Possibilité d'utiliser les rest et miss.

Tab. 6.5 – Comparaison de l'application des meilleures couvertures dans MKBEEM aux travaux les plus proches.

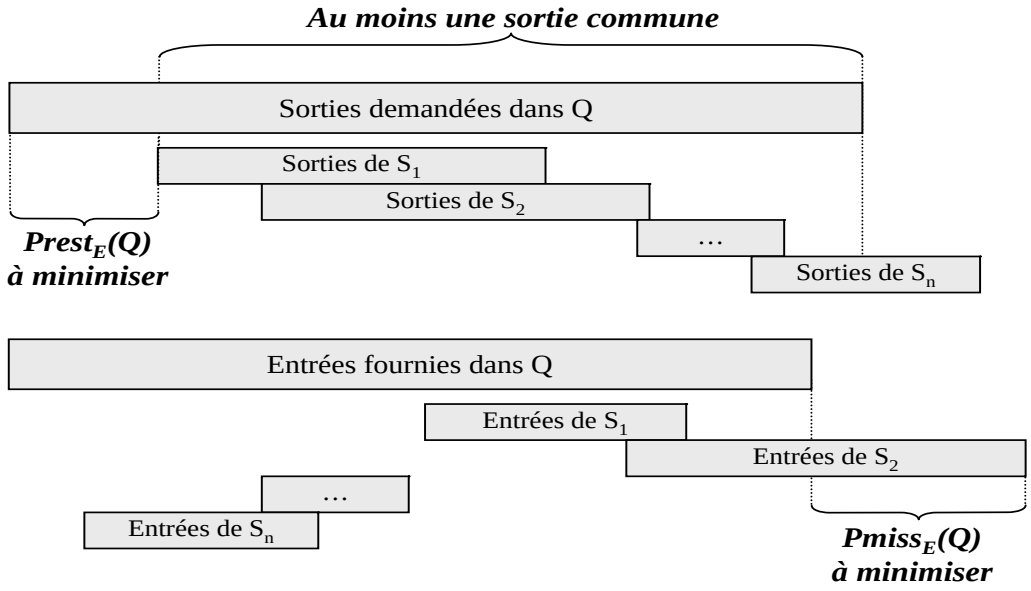


FIG. 6.4 – La découverte des meilleures couvertures de profile DAML-S.

- les combinaisons de services ayant au moins une sortie commune avec la requête sont des couvertures de profile de la requête,
- les couvertures ayant le plus possible de sorties communes et le moins possible d'entrées en plus par rapport à la requête sont des meilleures couvertures de profile de celle-ci (en d'autres termes le rest du profile, ou $Prest$, et le miss du profile, ou $Pmiss$, doivent être minimisés).

Le tableau 6.6 compare la découverte des meilleures couvertures de profile avec la découverte proposée dans [66] qui est la découverte de services DAML-S la plus proche. Les conclusions sont les suivantes : notre approche est plus flexible que [66] au prix de l'utilisation d'un langage ayant la propriété de subsomption structurelle donc un peu moins expressif que DAML-S.

Approche	• Correspondance avec Q • Définition • Langage	Degré de correspondance	Propriété intéressante
[66]	• des services S séparés • Chaque sortie de Q doit être mise en correspondance avec une sortie de S, et chaque entrée de S doit être mise en correspondance avec une sortie de Q. • DAML+OIL	Du meilleur au pire d'abord pour les sorties : – exact : $sorties(Q) \equiv sorties(S)$ – plug-in : $sorties(Q) \sqsubseteq sorties(S)$ – subsume : $sorties(Q) \sqsupseteq sorties(S)$ – échec si pas de relation de subsomption puis pour les entrées : – exact : $entrees(Q) \equiv entrees(S)$ – plug-in : $entrees(Q) \sqsupseteq entrees(S)$ – subsume : $entrees(Q) \sqsubseteq entrees(S)$ – échec si pas de relation de subsomption	Possibilité de fixer le degré de correspondance minimale.
Meill. couvertures de profile	• des combinaisons E de services • Au moins une sortie en commun entre Q et E : • Tous les sous-langages de DAML+OIL à subsomption structurelle	Minimisation successives de : 1) la taille de $Prest_E(Q)$: i.e. du meilleur au pire : – exact : $sorties(Q) \equiv sorties(E)$ – subsume : $sorties(Q) \sqsupseteq sorties(E)$ – plug-in : $sorties(Q) \sqsubseteq sorties(E)$ – si pas de relation de subsomption, E peut être acceptée si pas de E' meilleur. 2) la taille de $Pmiss_E(Q)$	Possibilité d'utiliser les rest et miss.

TAB. 6.6 – Comparaison des meilleures couvertures de profile avec la découverte de [66].

Chapitre 7

Implémentation

Sommaire

Dans ce chapitre, nous présentons les implémentations que nous avons réalisées concernant l'algorithme *computeBCov* ainsi que les tests que nous avons effectués. Nous rappelons que *computeBCov* est l'algorithme qui résoud la découverte des meilleures couvertures d'un concept étant donnée une terminologie pour un langage ayant la propriété de subsomption structurelle.

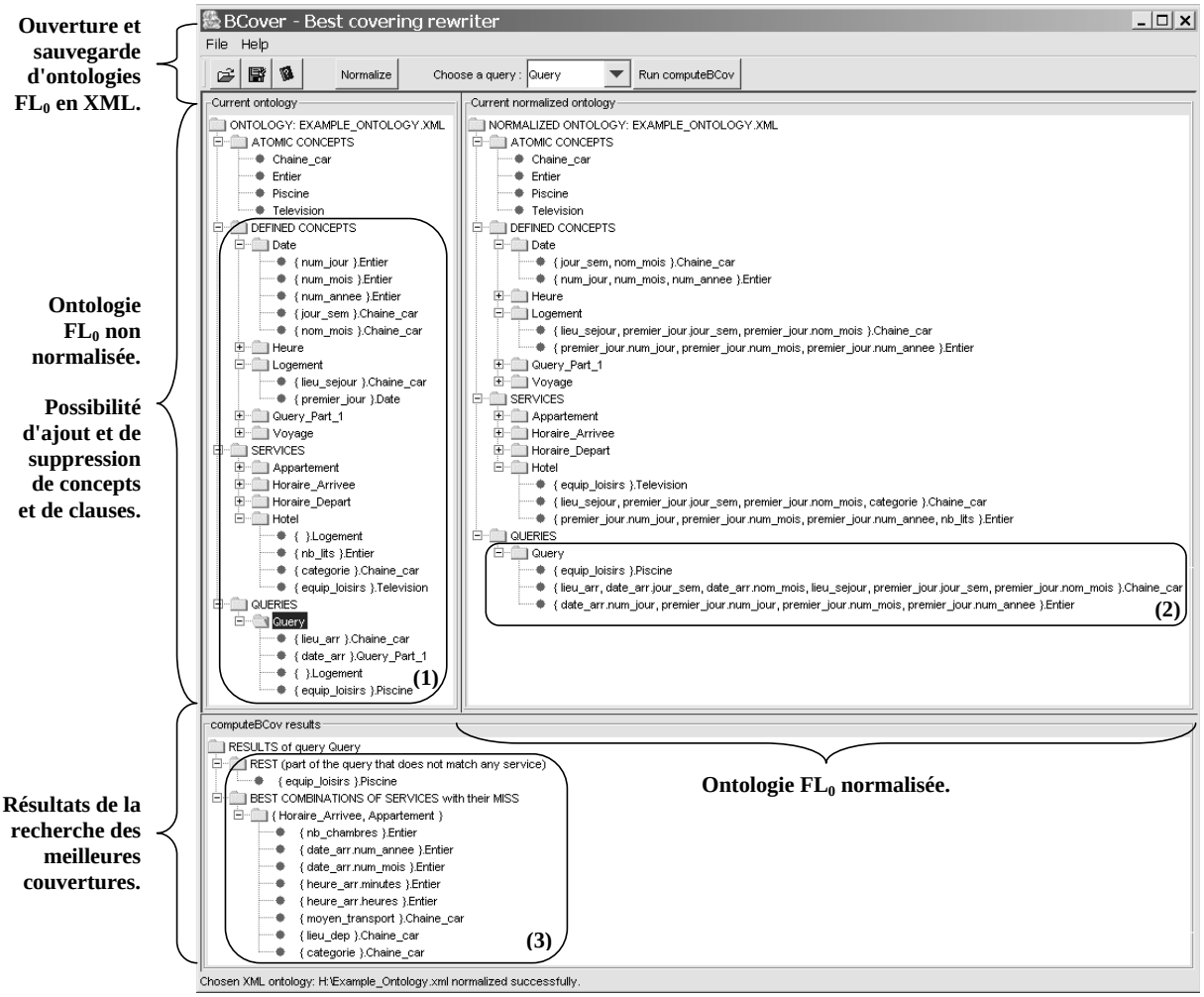
Dans un premier temps, nous avons implémenté *computeBCov* pour la logique de description $\mathcal{FL}_0$ au sein du projet MKBEEM. Puis, afin de mieux tester *computeBCov*, nous avons construit deux interfaces appelées BCover et D^2CP permettant de tester qualitativement et quantitativement *computeBCov*. La section 7.1 présente les fonctionnalités de BCover. La section 7.2 détaille quant à elle les fonctionnalités de D^2CP ainsi que les tests effectués avec *computeBCov* et ses variantes sur des ontologies générées aléatoirement. Ces tests permettent de vérifier l'efficacité des points importants de *computeBCov* (notamment les persistants et le Branch and Bound). Sans constituer un test substantiel du passage à l'échelle, ils montrent comment *computeBCov* se comporte sur quelques exemples, aléatoires, de taille relativement importante.

Comme nous avons utilisé *computeBCov* dans le cadre de la découverte dynamique de services, nous continuons dans ce chapitre à employer les termes "requête", "service" et "ontologie", au lieu de "concept à récrire", "concept défini" et "terminologie".

7.1 BCover

Dans cette section, nous présentons l'application Java nommée BCover dans laquelle nous avons testé qualitativement *computeBCov*. Cette application est essentiellement composée d'une interface de manipulation d'ontologies $\mathcal{FL}_0$ qui permet l'ajout et la suppression de définitions de concepts, de services et de requêtes, ainsi que l'exécution de ces requêtes avec *computeBCov*. La figure 7.1 montre l'interface de cette application et trace une exécution de *computeBCov* (cette exécution est celle de l'exemple de la section 6.2.1 page 108).

Grâce à cette application, nous avons construit des scenarii de découverte dans le contexte



Pour les descriptions correspondant à **(1)**, voir le tableau 6.2 page 109. Pour **(2)**, voir le tableau 6.3 page 110. Pour **(3)**, voir le tableau 6.4 page 111.

FIG. 7.1 – Interface de BCover sur l'exemple de la section 6.2.1.

de MKBEEM qui montrent l'intérêt de *computeBCov*. Le premier de ces scenarii est celui de l'exemple page 106. Les autres sont présentés à l'annexe E page 195. Tous ces cas permettent de vérifier par l'exemple les caractéristiques de la découverte vues précédemment (raisonnement flexible, réutilisation des rest et miss, recherche combinatoire).

7.2 D^2CP

Dans cette section, nous présentons l'application Java nommée D^2CP, pour "Dynamic Discovery of Concepts Prototype", soit "Prototype de Découverte Dynamique de Concepts", qui nous a permis de mener des tests quantitatifs de l'algorithme *computeBCov*. Le but principal de ces tests est de comprendre précisément l'influence de chaque mécanisme de *computeBCov* dans son comportement global (les persistants, le Branch and Bound et sa politique d'évaluation du coût). Nous vérifions notamment que le Branch and Bound et les persistants sont efficaces. Les tests de *computeBCov* réalisés sur des ontologies de tailles importantes donnent un aperçu des temps globaux d'exécution, et finalement de la faisabilité concrète de l'approche. Ces tests constituent un premier pas vers une étude du passage à l'échelle de *computeBCov* qui reste à faire.

7.2.1 Fonctionnalités de D^2CP

Le but de ce système est d'être une plate-forme de tests de l'algorithme *computeBCov*. Il permet de tester comparativement ses 6 variantes (selon les optimisations choisies) et de générer aléatoirement des ontologies $\mathcal{FL}_0$ qui peuvent être sauvegardées sous la forme de fichiers XML. Un module de D^2CP permet la visualisation des résultats sous diverses formes. La figure 7.2 résume ces fonctionnalités et la figure 7.3 montre l'interface graphique utilisateur de D^2CP.

Les 6 variantes de *computeBCov* sont définies par le choix d'utiliser ou non le théorème des persistants pour générer les transversaux minimaux (c'est-à-dire utiliser soit l'algorithme 2, soit l'algorithme 1), d'utiliser ou non le BnB pour trouver les transversaux de coût minimal et d'utiliser la politique 1 (BnB1) ou la politique 2 (BnB2) s'il utilise le BnB. Les six variantes de *computeBCov* sont résumées à la figure 7.5.

Le module de génération d'ontologies de D^2CP permet de générer des ontologies $\mathcal{FL}_0$ sous la forme de fichiers XML, à partir d'un fichier DTD qui décrit la structure de l'ontologie à générer. Les ontologies générées contiennent une description de la requête, un ensemble de descriptions de services et un ensemble de descriptions des autres concepts définis qui sont utilisés dans la description des services. La génération est divisée en deux étapes. La première est la génération d'un squelette d'ontologie, grâce au générateur IBM XML Generator[33], paramétré par un fichier DTD choisi par l'utilisateur et par des valeurs entrées via son l'interface graphique. La seconde est le renommage des éléments du squelette généré précédemment afin d'obtenir une ontologie $\mathcal{FL}_0$ sans cycle pouvant être traitée par *computeBCov*. Les paramètres de génération d'ontologie ajustables par l'utilisateur sont donnés au tableau 7.1.

L'utilisateur peut choisir le type d'affichage qu'il souhaite parmi un affichage textuel simple des meilleures combinaisons de services web, une trace arborescente de chaque exécution, didactique et utile pour vérifier le bon déroulement de chaque variante exécutée, et un affichage des statistiques d'exécution qui sont des mesures de chaque étape de *computeBCov* destinées à avoir une vision détaillée de l'exécution de chaque variante. Ces statistiques peuvent être présentées sous la forme d'un histogramme ou d'un tableau de valeurs (voir la figure 7.4), et sont par ailleurs

[33] Voir http ://www.alphaworks.ibm.com/tech/xmlgenerator.

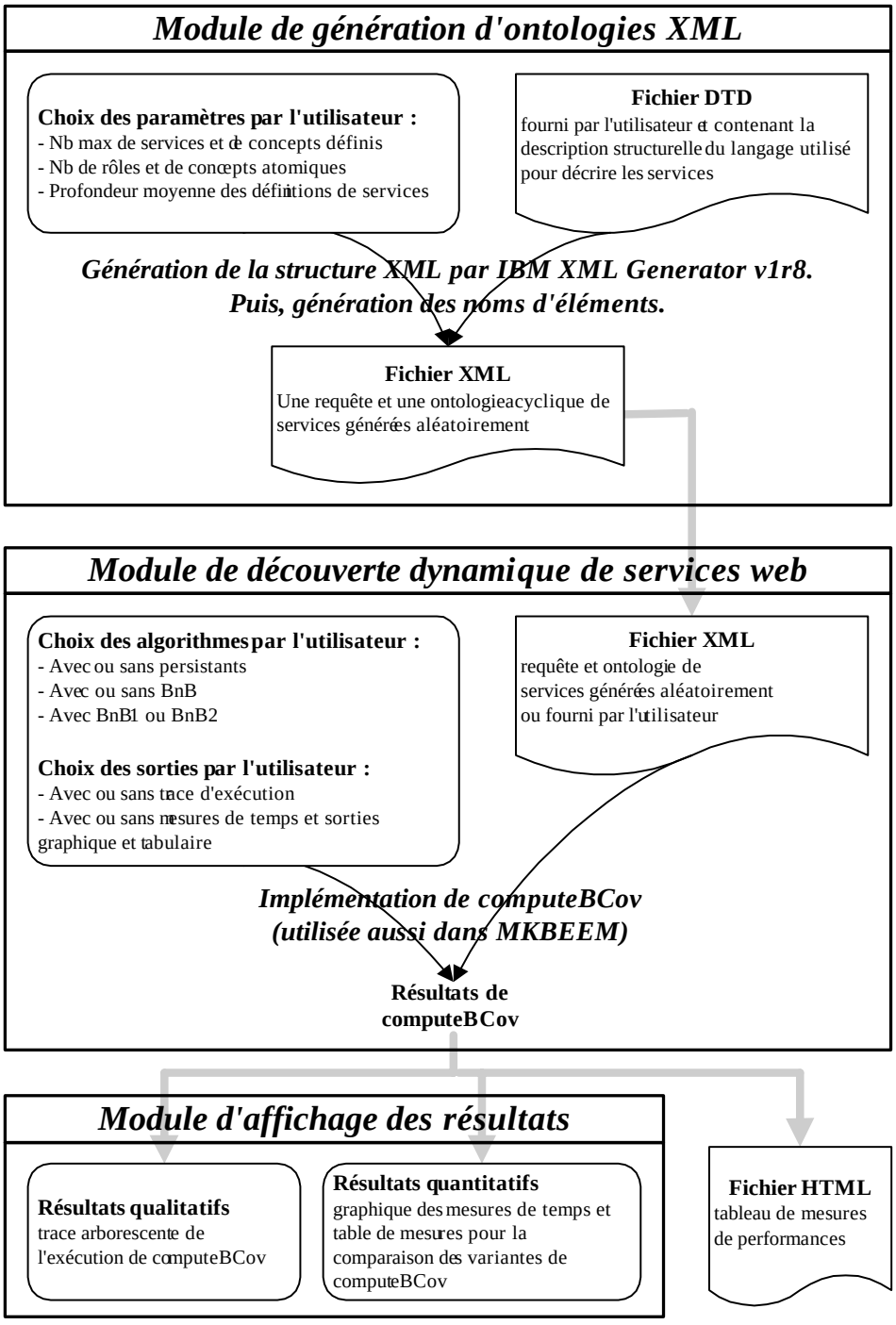


FIG. 7.2 – Vue d'ensemble des fonctionnalités de D^2CP.

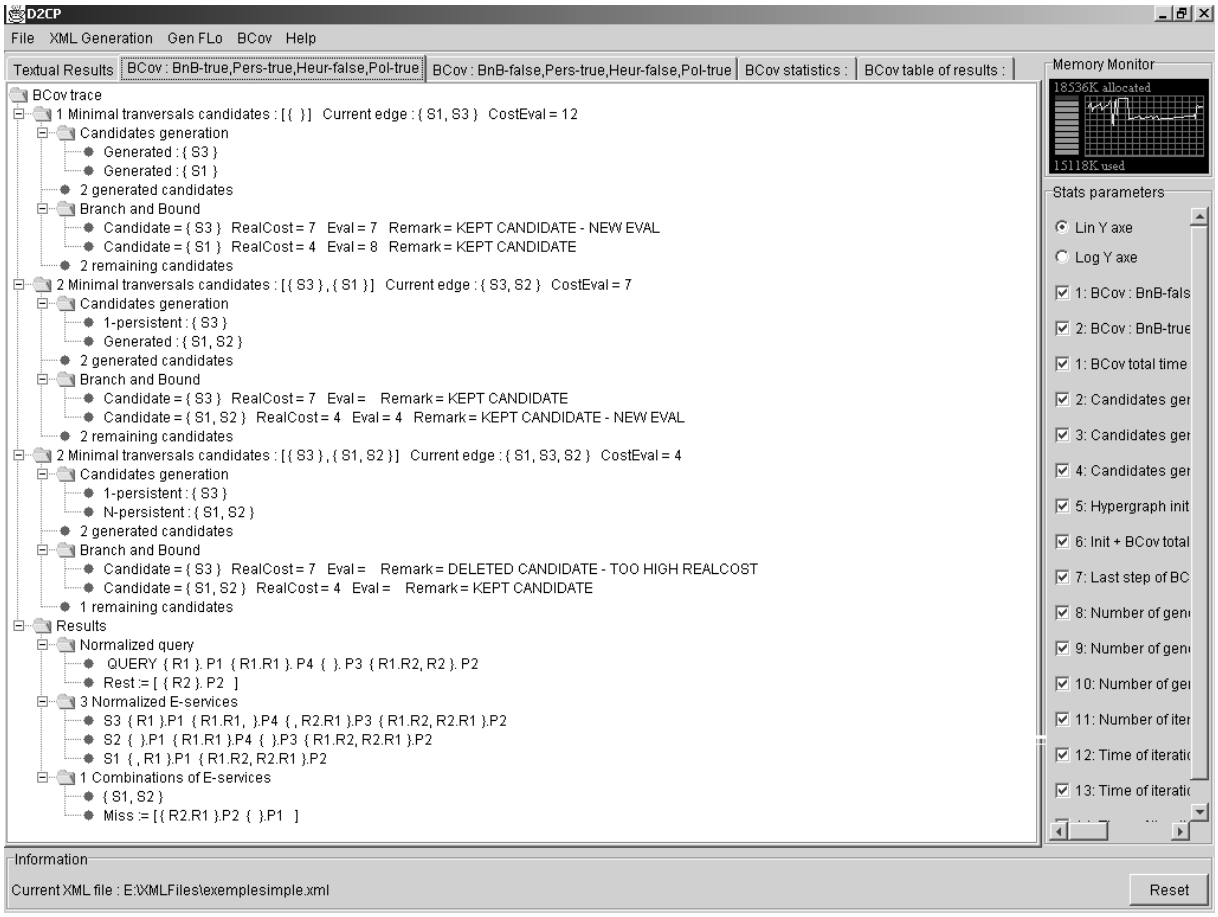


FIG. 7.3 – L'interface graphique de D^2CP. On peut voir la trace d'exécution de l'exemple détaillé dans la section 3.2.4 page 39.

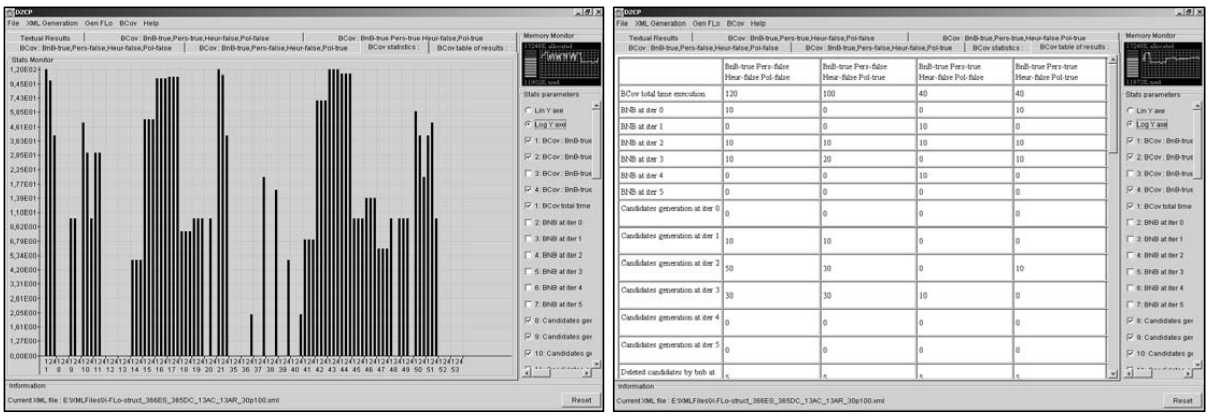


FIG. 7.4 – Les sorties de D^2CP sous forme graphique et tabulaire.

stockées dans un fichier HTML afin de servir d'entrée à un tableur. On trouvera dans [71, 70] une présentation de D^2CP.

7.2.2 Expérimentations avec D^2CP

Grâce au choix possible des variantes de *computeBCov* à exécuter et à la génération d'ontologies aléatoires, D^2CP nous a permis de réaliser des expérimentations quantitatives dont l'objectif est la comparaison des différentes variantes de *computeBCov* et donc la vérification de l'efficacité des optimisations proposées (persistants et Branch and Bound). Elles nous donnent aussi un aperçu du temps global d'exécution de *computeBCov*.

Les expérimentations sont menées en deux étapes. Dans un premier temps, nous avons testé *computeBCov* sur de petits exemples construits pour produire un nombre exponentiel de solutions en fonction du nombre de services présents dans l'ontologie et de la taille de la requête. Cette étude permet de vérifier la croissance exponentielle du temps d'exécution de *computeBCov* dans le pire des cas (essentiellement due aux calculs de transversaux minimaux eux-mêmes exponentiels). Dans un deuxième temps, nous avons généré aléatoirement des ontologies de plus grandes tailles pour étudier plus précisément le coût relatif des persistants, du Branch and Bound et de sa politique.

Expérimentations "au pire" de *computeBCov*

Nous reprenons ici le cas de l'hypergraphe H_2, construit dans l'annexe C.3, page 179, pour montrer l'évolution exponentielle de *computeBCov* dans de très mauvais cas. Il est simple de faire de H_2 une ontologie (de petite taille) associée à une requête, le tout étant une instance du problème de découverte des meilleures couvertures dans $\mathcal{FL}_0$. Ce cas est un très mauvais cas pour *computeBCov* car le nombre de solutions associées est exponentiel en fonction de sa taille, et le nombre d'opérations élémentaires est maximisé durant l'exécution de l'algorithme 2 pour la génération des transversaux minimaux. La figure 7.6 montre les résultats de l'exécution de *computeBCov* dans D^2CP (avec l'algorithme 2 pour générer les transversaux minimaux) sur plusieurs instances construites sur le modèle de H_2 mais avec des tailles croissantes. On a ainsi une idée de la borne supérieure pour le temps d'exécution total de *computeBCov* appliqué à des ontologies de petites tailles mais avec un grand nombre de solutions. Ainsi, on voit qu'il faut environ une seconde à *computeBCov* pour résoudre une instance du problème possédant jusqu'à 3264 solutions, et il faut entre 1 et 20 secondes pour résoudre une instance possédant jusqu'à 13056 solutions. Sachant que ces cas sont très mauvais pour l'algorithme 2, ces résultats sont encourageants.

Les différentes instances de H_2 ont comme caractéristique d'être de petits exemples (moins de 40 services), avec un grand nombre de solutions (jusqu'à 100 000, voir la figure 7.6). Or l'expérience de MKBEEM nous a montré que des exemples plus réalistes seraient plutôt de grands exemples (au moins 500 services et 2000 concepts définis) avec un petit nombre de solutions (moins de 10). De plus, ces cas rendent inutile le Branch and Bound puisqu'ils sont construits pour maximiser le nombre de solutions. Ainsi, ces cas sont a priori pires que des cas réalistes, et ne permettent pas de tester le Branch and Bound.

Premiers tests vers un passage à l'échelle

Il reste donc à tester *computeBCov* sur des exemples de plus grandes tailles avec un nombre réduit de solutions (i.e. plus réalistes en dimensions). Pour cela, on utilise le module de génération

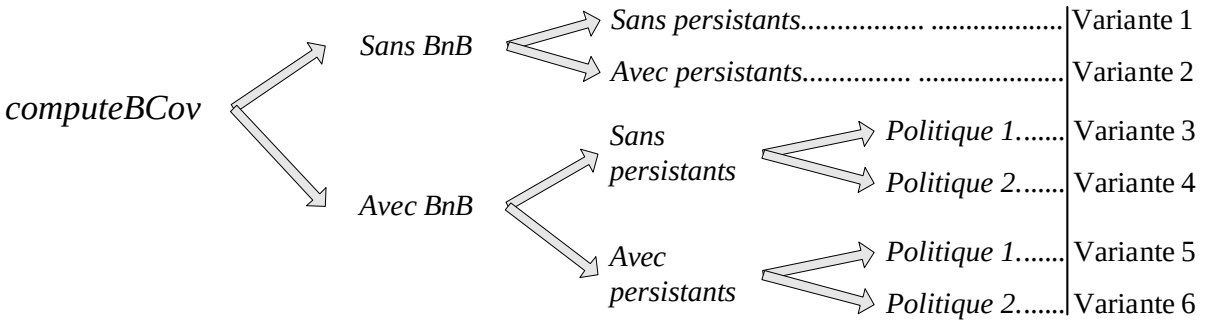


FIG. 7.5 – Les 6 variantes de *computeBCov*.

Nature du paramètre
Le nombre maximal de définitions de concepts (services et autres).
La proportion de définitions de services par rapport aux autres définitions de concepts.
Le nombre maximal de clauses dans les descriptions de concepts (i.e. le nombre maximal d'éléments fils de l'élément *AND* dans le fichier XML).
La proportion d'éléments *Atomic-Concept* par rapports aux éléments *FORALL* dans chaque description de concept.
Le nombre total de concepts atomiques utilisés dans l'ontologie.
Le nombre total de rôles atomiques utilisés dans l'ontologie.
La proportion de concepts définis (non services) réutilisés pour définir d'autres concepts (services ou non services).

TAB. 7.1 – Paramètres réglables pour la génération d'ontologies XML dans D^2CP

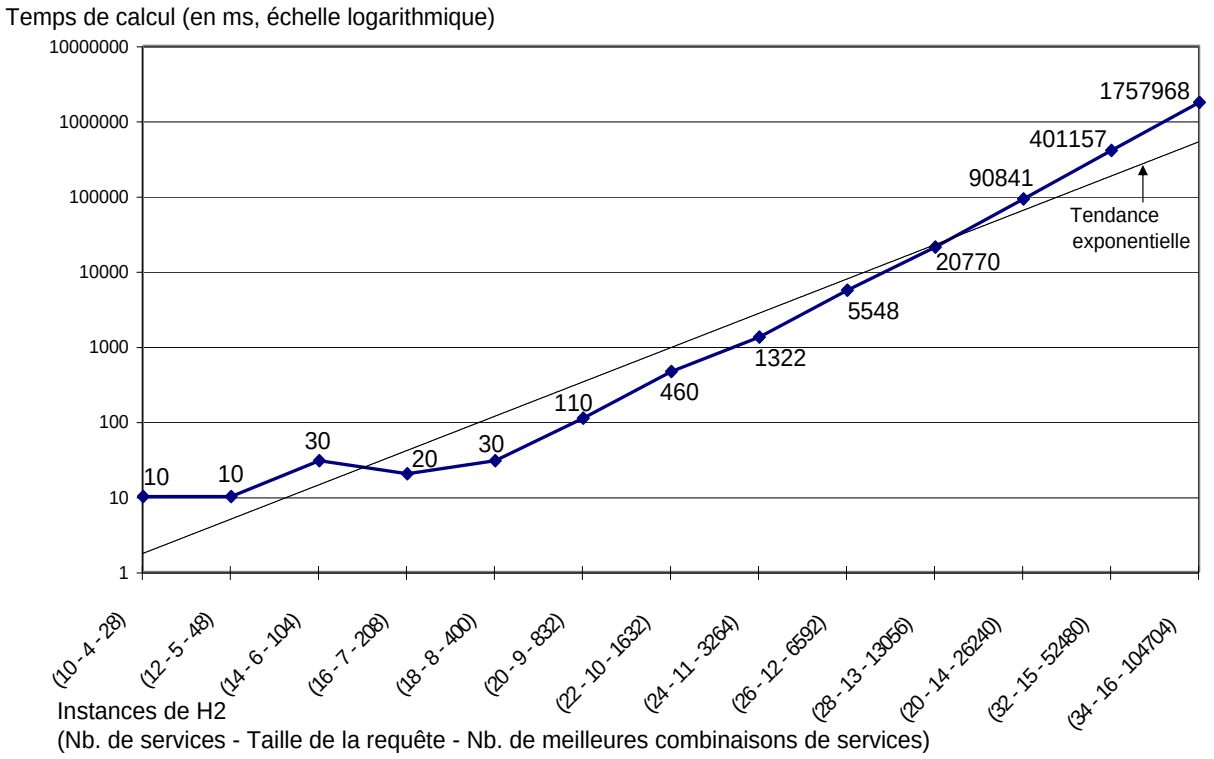


FIG. 7.6 – Temps total d'exécution de *computeBCov* (avec l'algorithme 2) appliqué à des ontologies et des requêtes de petites tailles mais résultant en un nombre exponentiel de meilleures combinaisons de services en fonction de ces tailles d'entrées. Pour x le nombre de services, y la taille de la requête et z le nombre de solutions, on a environ $z = 2^{(x+y)/3}$.

aléatoire d'ontologies de D^2CP. Ces exemples permettent d'étudier l'effet du Branch and Bound (à la fois le gain et le coût qu'il implique).

Parmi toutes les ontologies que nous avons générées grâce à D^2CP, nous en avons selectionné trois qui sont représentatives de tous les cas de figure que l'on a pu observer. Après avoir présenté ces trois cas, nous discutons des temps d'exécution de chaque variante sur chacun des cas.

Les caractéristiques de ces trois cas sont résumées au tableau 7.2. Pour les étudier, nous considérons une « ontologie repère » qui représente une ontologie possédant des caractéristiques qui nous semblent se rapprocher d'une ontologie réaliste[34]. Pour établir si le cas est favorable ou non par rapport à *computeBCov*, on raisonne sur les nombres de concepts et rôles atomiques : plus ces valeurs sont petites, plus ces concepts et rôles atomiques auront été réutilisés durant la génération aléatoire. Ceci implique une combinatoire plus grande, et donc une découverte dynamique plus lente a priori.

Les temps globaux d'exécution des 6 variantes de *computeBCov* dans D^2CP sont donnés à la figure 7.7. Celle-ci montre que, pour les cas 1 et 3, il existe au moins une variante qui effectue la découverte dynamique en moins de deux secondes, et en 30 secondes pour le cas 2. Ces tests montrent aussi qu'il y a une grande différence de performance suivant la variante utilisée. La figure 7.7 ainsi que l'annexe D qui étudie dans le détail les trois ontologies, nous permettent de tirer les conclusions suivantes :

[34]Ces valeurs prennent en compte l'expérience acquise avec le système MKBEEM. Elles restent arbitraires et sujettes à ajustements.

Caractéristique	Onto. repère	Cas 1	Cas 2	Cas 3
Nombre de concepts définis	2500	365	1334	3405
Nombre de services (i.e. de sommets dans l'hypergraphe)	500	366	660	570
Nombre de clauses dans la requête (i.e. d'arêtes dans l'hypergraphe)	15	6	33	12
Nombre moyen de services pour une clause (i.e. de sommets dans une arête)	20	10.83	20.84	30.75
Nombre de concepts atomiques	150	13	30	12
Nombre de rôles	100	13	30	12
Proportion de concept définis utilisés dans d'autres définitions de concepts	30%	30%	20%	33%

Ontologie	Description
Cas 1	Assez différente de l'ontologie repère de par sa petite taille, petite requête, cas très défavorable pour *computeBCov*.
Cas 2	Proche de l'ontologie repère mais de petite taille, requête de taille normale, cas défavorable pour *computeBCov*.
Cas 3	Très proche de l'ontologie repère, petite requête, cas très défavorable pour *computeBCov*.

TAB. 7.2 – Caractéristiques de l'ontologie repère et des 3 cas détudes générés par D^2CP.

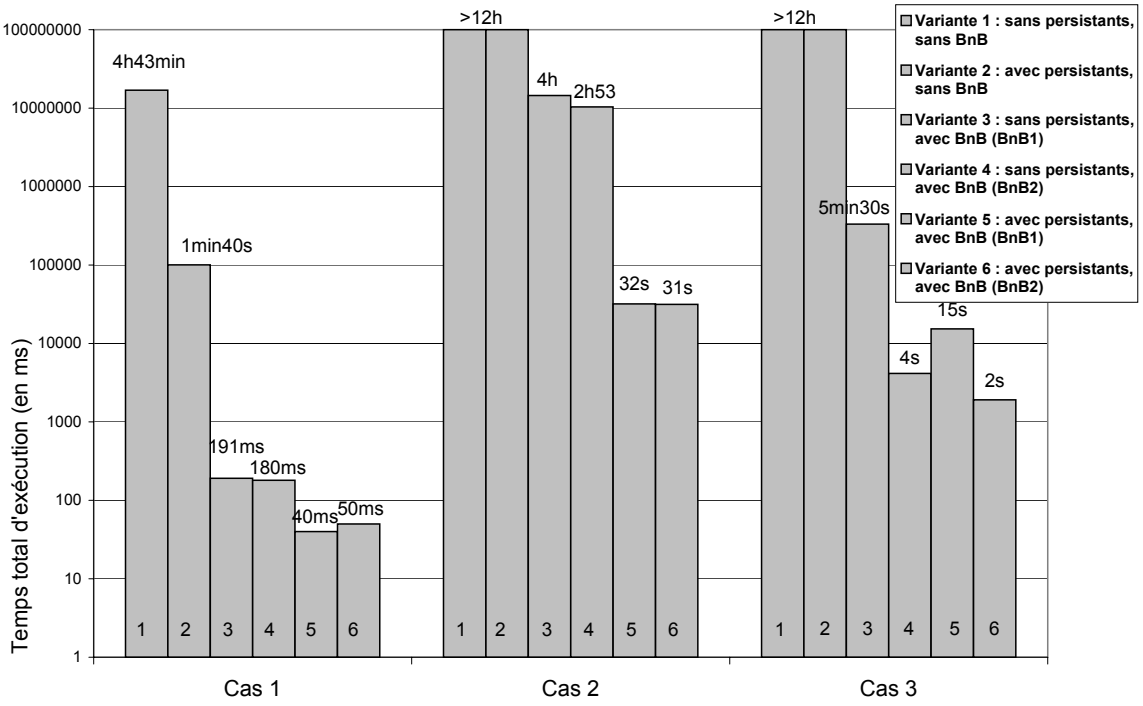


FIG. 7.7 – Temps global d'exécution, en millisecondes, de chaque variante de *computeBCov* pour les 3 cas d'étude dans D^2CP

- l'optimisation par le Branch and Bound est très efficace : pour le cas 1, même sans les persistants, le BnB permet d'obtenir une solution en temps réel (moins d'une seconde). Pour les cas 2 et 3, le BnB permet d'envisager l'obtention d'une solution dans un temps limité sans les persistants, voire presque en temps réel s'il est associé aux persistants. Sans le BnB, ces cas sont intraitables (plus de 12 heures). L'effet espéré du Branch and Bound, qui est de limiter l'explosion combinatoire du problème, est donc bien présent. Cet effet est encore accentué par l'usage de BnB2 au lieu de BnB1 : comme BnB2 permet une meilleure évaluation d'une solution possible à chaque itération, le BnB implémenté avec BnB2 élague plus de branches dans l'arbre des combinaisons ;
- les persistants sont aussi très intéressants, mais uniquement lorsqu'ils sont associés avec le BnB. Ils n'ont pas d'effet sur le nombre de combinaisons explorées, et donc ne limitent pas l'explosion combinatoire. Cependant les persistants accélèrent très sensiblement la génération des combinaisons à explorer. Dans le cas 2, associés avec le BnB, ils permettent l'obtention des solutions en un temps raisonnable (environ 30 secondes). Dans le cas 3, ils permettent l'obtention des solutions presque en temps réel (quelques secondes).

Ainsi, la variante la plus performante de *computeBCov* est celle qui associe le BnB implémenté avec la méthode BnB2 et les persistants. C'est cette variante qui a été utilisée pour valider la découverte de services dans MKBEEM. Les cas étudiés étant de tailles respectables, proches de l'ontologie repère et de nature assez défavorable pour *computeBCov*, on peut être optimiste quant aux performances de D^2CP sur des ontologies réelles qui restent à construire.

En conclusion, le tableau 7.3 résume les apports du BnB, des persistants, et de la politique 2 dans *computeBCov*.

Mécanisme de *computeBCov*	Principal effet
Branch and Bound	Limite l'explosion combinatoire dans le nombre de siolutions candidates examinées.
Politique 2 du Branch and Bound	Accentue l'efficacité du Branch and Bound.
Persistants	Diminue le temps moyen de génération d'un candidat.

TAB. 7.3 – Apport du BnB, des persistants et de la politique 2 dans *computeBCov*.

Conclusion

Dans cette thèse, nous avons étudié le problème de la découverte des meilleures couvertures d'un concept en utilisant une terminologie. Ce problème constitue une nouvelle instance du cadre formel de la réécriture de concept en utilisant une terminologie pour les logiques de description. Sa principale caractéristique est d'être basée sur le calcul de la différence sémantique entre le concept à récrire et ses réécritures, offrant ainsi une flexibilité plus grande que les réécritures existantes. Ce nouveau raisonnement des logiques de description a été appliqué au problème de la découverte de services web sémantiques.

Nous avons étudié le problème de la découverte des meilleures couvertures pour les langages ayant la propriété de subsomption structurelle, et également pour le langage $\mathcal{ALN}$.

Dans le cas des logiques de description ayant la propriété de subsomption structurelle, selon la définition de [74], la différence est sémantiquement unique et facile à calculer. Il est apparu que rechercher les meilleures couvertures pour ces langages revenait à rechercher les transversaux minimaux de coût minimal dans un hypergraphe construit à partir du concept à couvrir et de la terminologie. Nous avons montré que ce problème était NP-Difficile, et nous avons proposé un algorithme nommé *computeBCov* pour le résoudre.

Dans le cas de la logique de description $\mathcal{ALN}$, la possibilité d'exprimer l'inconsistance de manière non triviale rend la découverte des meilleures couvertures plus complexe que dans le premier cas. En particulier, la différence n'est plus sémantiquement unique, et la notion de meilleure couverture doit être redéfinie. Nous avons proposé une méthode pour calculer cette différence non sémantiquement unique et nous avons montré que le problème de la découverte des meilleures couvertures, redéfini dans $\mathcal{ALN}$, est NP-Difficile. Nous avons expliqué en quoi cette réécriture généralisait les réécritures existantes. En termes algorithmiques, la présence d'inconsistances non triviales implique que la découverte des meilleures couvertures est plus complexe que la réécriture classique (maximalement contenue). En particulier les approches habituelles par paniers ne sont plus adaptables. Ainsi, nous avons proposé un algorithme nommé *computeALNBCov* pour découvrir les meilleures couvertures dans $\mathcal{ALN}$. Basé sur le calcul exhaustif des cas d'inconsistances entre la terminologie et le concept à couvrir, cet algorithme consiste en des réductions successives de l'espace de recherche (le treillis des parties de l'ensemble des concepts de la terminologie). C'est une approche algorithmique nouvelle dans le domaine des réécritures.

En termes applicatifs, nous avons présenté l'utilisation des meilleures couvertures pour la découverte dynamique de services web sémantiques. En permettant la découverte selon un critère flexible d'ensembles de services pour une requête donnée, la découverte des meilleures couvertures représente un raisonnement original bien adapté à ce problème. De plus, nous avons montré qu'elle était plus flexible que les découverte existantes, bien que concernant des langages moins expressifs.

Les implémentations réalisées sont les suivantes : dans le cadre du projet européen MK-BEEM, un module de découverte de services web sémantiques implémentant *computeBCov* a été conçu et implanté ; deux autres programmes, nommés respectivement BCover et D^2CP, ont été développés pour tester qualitativement et quantitativement le fonctionnement de *computeBCov*.

Diverses perspectives peuvent être envisagées. Dans le domaine des logiques de description, une première extension de ce travail serait l'étude des meilleures couvertures pour un langage plus expressif que ceux ayant la propriété de subsomption structurelle et qu'$\mathcal{ALN}$. Le but serait de se rapprocher des langages très expressifs utilisés et normalisés dans le domaine du web sémantique. Cependant, les nombreuses interactions nouvelles entre les constructeurs de ces langages laissent à penser que les configurations d'inconsistances au sein de conjonctions de descriptions seront beaucoup plus nombreuses et donc que la complexité totale de la découverte des meilleures couvertures n'en sera que plus élevée. De plus, la présence de constructeurs tels que la disjonction et la négation complète remettent en cause la pertinence de l'opérateur de différence sémantique tel qu'il a été défini par Teege. Etudier les meilleures couvertures avec ces constructeurs impliquerait au préalable de redéfinir la différence sémantique en essayant de conserver les principes de la différence de Teege. Cette nouvelle définition plus générale de la différence sémantique, et l'étude associée, constituent une seconde extension possible à cette thèse dans le domaine des logiques de description.

D'un point de vue algorithmique, il serait intéressant de réaliser une étude comparative approfondie du fonctionnement et de la complexité des approches existantes de calcul des transversaux minimaux d'un hypergraphe. L'intérêt serait d'obtenir des valeurs de complexité plus précises pouvant nous permettre d'affiner l'étude de complexité de *computeALNBCov* et d'optimiser son implémentation.

En ce qui concerne la découverte de services, on a vu que l'on pouvait utiliser les meilleures couvertures et notamment le calcul des rest associés dans un contexte pair-à-pair, pour propager la partie de la requête qui n'avait pas été comprise par un pair et éventuellement compléter sa réponse. Dans la lignée de travaux actuels qui étudient les problématiques de l'intégration de données, de la médiation de schémas ou de la découverte de services web en contexte pair-à-pair, il serait intéressant de prolonger l'étude des meilleures couvertures dans ce contexte. En particulier, la sémantique des liens entre pairs, les politiques de transmission des rest d'un pair à un autre et d'arrêt de leur propagation nécessiteraient d'être formalisées.

Un des principaux enjeux des services web est la composition dynamique de services en des services plus complexes créés automatiquement à partir de services plus simples. Dans cette optique, la découverte de services proposée dans cette thèse peut être considérée comme la première étape de la composition dynamique. En effet, les meilleures couvertures permettent de découvrir des combinaisons de services. Orchestrer les services de ces combinaisons les uns par rapport aux autres, c'est-à-dire organiser leur succession dans le temps, serait une première approche de composition dynamique. Ceci apparaît d'autant plus envisageable avec la découverte des meilleures couvertures de profiles puisque la découverte s'effectue alors sur les entrées et sorties des services, éléments déterminants si l'on veut enchaîner leur exécution.

Enfin, il reste à implémenter *computeALNBCov* et à le tester quantitativement et qualitativement. Au vu des résultats de complexité, des optimisations sont sans doute nécessaires. Organiser l'algorithme de manière à ne pas faire deux fois les mêmes calculs apparaît incontournable. En particulier on peut se demander si certains des résultats des calculs (notamment des

calculs d'inconsistances) peuvent être conservés en vue de prendre en compte plus efficacement des variations dans la requête et les services.

Bibliographie

[1] A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. McDermott, D. Martin, S. A. McIlraith, S. Narayanan, M. Paolucci, and T. Payne. Daml-s : Web service description for the semantic web. In *Proceedings of the 1st Int'l Semantic Web Conf. (ISWC 02)*, 2002.

[2] F. Baader. A graph-theoretic generalization of the least common subsumer and the most specific concept in the description logic el. LTCS-Report LTCS-04-02, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2004. See http ://lat.inf.tu-dresden.de/research/reports.html.

[3] F. Baader, I. Horrocks, and U. Sattler. Description logics as ontology languages for the semantic web. In Dieter Hutter and Werner Stephan, editors, *Festschrift in honor of Jörg Siekmann*, Lecture Notes in Artificial Intelligence. Springer, 2003. To appear.

[4] F. Baader, R. Küsters, A. Borgida, and D. McGuinness. Matching in description logics. *Journal of Logic and Computation*, 9(3) :411–447, 1999.

[5] F. Baader, R. Küsters, and R. Molitor. Structural subsumption considered from an automata theoretic point of view. LTCS-Report LTCS-98-04, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 1998. See http ://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[6] F. Baader, R. Küsters, and R. Molitor. Computing Least Common Subsumer in Description Logics with Existential Restrictions. In T. Dean, editor, *Proceedings of IJCAI'99*, pages 96–101. mk, 1999.

[7] F. Baader, R. Küsters, and R. Molitor. Revised version of ltcs-report 99-12 : Rewriting concepts using terminologies - revisited. Technical report, Aachen University of Technology, Research group for Theoretical Computer Science, 2000.

[8] F. Baader, R. Küsters, and R. Molitor. Rewriting concepts using terminologies. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *Proceedings of the Seventh International Conference on Knowledge Representation and Reasoning (KR2000)*, pages 297–308, San Francisco, CA, 2000. Morgan Kaufmann Publishers.

[9] F. Baader, R. Küsters, and R. Molitor. Rewriting concepts using terminologies – revisited. Report 00-04, LTCS, RWTH Aachen, Germany, 2000. See http ://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[10] K. Baina, B. Benatallah, H.-Y. Paik, F. Toumani, C. Rey, A. Rutkowska, and B. Harianto. WS-CatalogNet : An infrastructure for creating, peering, and querying e-catalog communities. In *Proceedings of VLDB 2004, 30th International Conference on Very Large Data Bases, Toronto, Canada - Demonstration*, September 2004.

[11] S. Bansal and J. M. Vidal. Matchmaking of web services based on the daml-s service model. In *Proceedings of the second international joint conference on Autonomous agents*

and multiagent systems, POSTER SESSION, Melbourne, Australia, pages 926 – 927. ACM Press New York, NY, USA, 2003.

[12] C. Beeri, A.Y. Levy, and M.-C. Rousset. Rewriting queries using views in description logics. In L. Yuan, editor, *Proceedings of the 16th ACM SIG-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'97)*, pages 99–108, 1997.

[13] B. Benatallah, M.-S. Hacid, H. Paik, C. Rey, and F. Toumani. Peering and querying e-catalog communities. 2004. ICDE2004, Poster session.

[14] B. Benatallah, M.-S. Hacid, H.-Y. Paik, C. Rey, and F. Toumani. Towards semantic-driven, flexible and scalable framework for peering and querying e-catalog communities. *Information Systems, special issue on semantic web. A paraître.*, (117).

[15] B. Benatallah, M.-S. Hacid, C. Rey, and F. Toumani. Request rewriting-based web service discovery. In *Proceedings of the Second International Semantic Web Conference (ISWC)*, volume 2870 of *LNCS*, pages 242–257, Sanibel Island, FL, USA, October 2003. Springer.

[16] C. Berge. *Hypergraphs*, volume 45 of *North Holland Mathematical Library*. Elsevier Science Publishers B.V. (North-Holland), 1989.

[17] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[18] A. Bernstein and M. Klein. Discovering services : Towards high-precision service retrieval. In *Proceedings of the Web Services, E-Business, and the Semantic Web, CAiSE 2002 International Workshop, WES 2002, Toronto, Canada, May 27-28, 2002, Revised Papers*, volume 2512 of *Lecture Notes in Computer Science*, pages 260–276. Springer, 2002.

[19] A. Borgida and P.F. Patel-Schneider. A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic. *jair*, 1 :277–308, may 1994.

[20] R.J. Brachman and H.J. Levesque. The Tractability of Subsumption in Frame Based Description Languages. In *Proceedings of the 4th National Conference on Artificial Intelligence AAAI '84, Austin, USA*, pages 34–37, 1984.

[21] S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. Technical Report LTCS-Report 01-06, Aachen University of Technology, Research Group for Theoretical Computer Science, June 2001.

[22] S. Brandt, R. Küsters, and A.-Y. Turhan. Approximating $\mathcal{ALCN}$-Concept Descriptions. In *Proceedings of the International Workshop on Description Logics 2002 (DL 2002)*, 2002. Available from http ://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/.

[23] S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and Difference in Description Logics. In *Proceedings of the Eight International Conference on Knowledge Representation and Reasoning (KR2002)*, 2002.

[24] F. Casati and M.-C. Shan. Dynamic and adaptive composition of e-services. *Information Systems*, 26(3) :143–163, May 2001.

[25] F. Casati and M.-C. Shan. Models and Languages for Describing and Discovering E-Services. In *Proceedings of SIGMOD 2001, Santa Barbara, USA*, May 2001.

[26] D. Chakraborty, F. Perich, S. Avancha, and A. Joshi. Dreggie : Semantic service discovery for m-commerce applications, 2001.

[27] W. Cohen, A. Borgida, and H. Hirsh. Computing least common subsumers in description logics. In Paul Rosenbloom and Peter Szolovits, editors, *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 754–761, Menlo Park, California, 1992. AAAI Press.

[28] O. Corcho, A. Gomez-Perez, A. Léger, C. Rey, and F. Toumani. An ontology-based mediation architecture for e-commerce applications. In *Proceedings of Intelligent Information Systems 2003*, Zakopane, Poland, June 2003.

[29] A. Dogac, I. Cingil, G. Laleci, and Y. Kabak. Improving the functionality of uddi registries through web service semantics. In *Technologies for E-Services, Third International Workshop, TES 2002, Hong Kong, China, August 23-24, 2002, Proceedings*, volume 2444 of *Lecture Notes in Computer Science*, pages 9–18. Springer, 2002.

[30] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In *In Gerhard Brewka, editor, Foundation of Knowledge Representation*, pages 191–236. CSLI-Publications, 1996.

[31] O.M. Duschka. Query optimization using local completeness. In *Proceedings of the Fourteenth AAAI National Conference on Artificial Intelligence, AAAI-97.*, 1997.

[32] T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM Journal on Computing*, 24(6) :1278–1304, 1995.

[33] T. Eiter and G. Gottlob. Hypergraph transversal computation and related problems in logic and ai. In Sergio Flesca, Sergio Greco, Nicola Leone, and Giovambattista Ianni, editors, *Proceeding of the Logics in Artificial Intelligence, European Conference, JELIA, Cosenza, Italy, September, 23-26*, volume 2424 of *Lecture Notes in Computer Science*. Springer, 2002.

[34] T. Eiter, G. Gottlob, and K. Makino. New results on monotone dualization and generating hypergraph transversals. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC-02)*, pages 14–22, New York, may19–21 2002. ACM Press.

[35] D. Fensel, C. Bussler, and A. Maedche. Semantic Web Enabled Web Services. In *International Semantic Web Conference, Sardinia, Italy*, pages 1–2, jun 2002.

[36] D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, and P. F. Patel-Schneider. Oil : An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2) :38–45, 2001.

[37] M.L. Fredman and L. Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21(3) :618–628, November 1996.

[38] M. Garey and D.S. Johnson. *Computers and Intractability - a Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979.

[39] F. Goasdoué. *Réécriture de requêtes en termes de vues dans CARIN et intégration d'informations*. PhD thesis, Université Paris XI Orsay, novembre 2001.

[40] F. Goasdoué, V. Lattès, and M.-C. Rousset. The Use of CARIN Language and Algorithms for Information Integration : The PICSEL System. *IJICIS*, 9(4) :383–401, 2000.

[41] F. Goasdoué and M.-C. Rousset. Compilation and approximation of conjunctive queries by concept descriptions. In Ian Horrocks and Sergio Tessaris, editors, *Proceedings of the 2002 International Workshop on Description Logics (DL2002)*, pages 221–232, 2002.

[42] F. Goasdoué and M.-C. Rousset. Answering queries using views : a krdb perspective for the semantic web. *ACM Journal - Transactions on Internet Technology (TOIT)*, 2003.

[43] J. González-Castillo, D. Trastour, and C. Bartolini. Description Logics for Matchmaking of Services. In *Proc. of the KI-2001 Workshop on Applications of Description Logics Vienna, Austria*, volume 44, September 2001.

[44] A.Y. Halevy. Answering queries using views : A survey. *VLDB Journal*, 2001.

[45] I. Horrocks. DAML+OIL : a reason-able web ontology language. In *Proceedings of the Advances in Database Technology - EDBT 2002, 8th International Conference on Extending Database Technology, Prague, Czech Republic, March 25-27*, volume 2287 of *Lecture Notes in Computer Science*. Springer, 2002.

[46] I. Horrocks, P.F.Patel-Schneider, and F . van Harmelen. Reviewing the Design of DAML+OIL : An Ontology Language for the Semantic Web. In *Proc. of the 18th Nat. Conf. on Artificial Intelligence (AAAI 2002)*, 2002. To appear.

[47] D.J. Kavvadias and E.C. Stavropoulos. Evaluation of an algorithm for the transversal hypergraph problem. *Lecture Notes in Computer Science*, 1668 :72–85, 1999.

[48] M. Klein and A. Bernstein. Searching for services on the semantic web using process ontologies. In Isabel Cruz, Stefan Decker, Jérôme Euzenat, and Deborah McGuinness, editors, *The Emerging Semantic Web - Selected papers from the first Semantic Web Working Symposium*, pages 159–172. IOS press, Amsterdam, 2002.

[49] R. Küsters. Characterizing the semantics of terminological cycles in ALN using finite automata. LTCS-Report LTCS-97-04, LuFg Theoretical Computer Science, RWTH Aachen, Germany, 1997.

[50] R. Küsters. *Non-Standard Inferences in Description Logics*, volume 2100 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2001. Ph.D. thesis.

[51] R. Küsters and R. Molitor. Computing least common subsumers in alen. LTCS-Report 00-07, LTCS, RWTH Aachen, Germany, 2000. See http ://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[52] R. Küsters and R. Molitor. Computing Least Common Subsumers in ALEN. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 219–224. Morgan Kaufman, 2001.

[53] A. Léger, G. Michel, P. Barrett, S. Gitton, A. Goméz-Pérez, A. Lehtola, K. Mokkila, S. Rodrigez, J. Sallantin, T. Varvarigou, and J. Vinesse. Ontology domain modeling support for multi-lingual services in e-commerce : MKBEEM. In *Proceedings of the Workshop on Applications of Ontologies and Problem Solving Methods, 14th European Conference on Artificial Intelligence ECAI'00, Berlin, Germany*, August 2000.

[54] M. Lenzerini. Data integration : A theoretical perspective. In *Proceedings of the Twenty-First Symposium on Principles of Database Systems (PODS 2002), June 3-5, Madison, Wisconsin, USA*. ACM, 2002.

[55] A. Levy, A. Rajaraman, and J. Ordille. Querying heterogeneous information sources using source descriptions. In T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, and Nandlal L. Sarda, editors, *Proceedings of the 22nd VLDB Conference, Bombay, India*, pages 251–262. Morgan Kaufmann, sep 1996.

[56] A.Y. Levy. Obtaining complete answers from incomplete databases. In T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, and Nandlal L. Sarda, editors, *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, pages 402–412. Morgan Kaufmann, 1996.

[57] A.Y. Levy and M.-C. Rousset. CARIN : A representation language combining horn rules and description logics. In *European Conference on Artificial Intelligence*, pages 323–327, 1996.

[58] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the Twelfth International World Wide Web Conference (WWW'2003)*, 2003.

[59] S. Lopes, J.-M. Petit, and L. Lakhal. Efficient discovery of functional dependencies and armstrong relations. In C. Zaniolo, P.C. Lockemann, M.H. Scholl, and T. Grust, editors, *Proceedings of the 7th International Conference on Extending Database Technology (EDBT 2000), Konstanz, Germany*, volume 1777 of *Lecture Notes in Computer Science*, pages 350–364. Springer, 2000.

[60] C. Lutz and U. Sattler. A proposal for describing services with DLs. In *Proceedings of the 2002 International Workshop on Description Logics*, 2002.

[61] H. Mannila and K-J Räihä. *The Design of Relational Databases*. Addison-Wesley, Wokingham, England, 1994.

[62] S. McIlraith, T. Son, and H. Zeng. Semantic web services, 2001.

[63] R. Molitor. Structural subsumption for $\mathcal{ALN}$. Technical Report LTCS-98-03, Aachen University of Technology, Research Group for Theoretical Computer Science, March 1998.

[64] D. Nardi, R. J. Brachman, F. Baader, W. Nutt, F. M. Donini, U. Sattler, D. Calvanese, R. Mölitor, G. De Giacomo, R. Küsters, F. Wolter, D. L. McGuinness, P. F. Patel-Schneider, R. Möller, V. Haarslev, I. Horrocks, A. Borgida, C. Welty, A. Rector, E. Franconi, M. Lenzerini, and R. Rosati. *The Description Logic Handbook. Theory, Implementation and Applications*. Cambridge University Press. F. Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi and Peter Patel-Schneider editors, January 2003.

[65] B. Nebel. Terminological Reasoning is Inherently Intractable. *ai*, 43 :235–249, 1990.

[66] M. Paolucci, T. Kawamura, T.R. Payne, and K.P. Sycara. Semantic Matching of Web Services Capabilities. In *Proc. of the Int. Semantic Web Conference, Sardinia, Italy*, pages 333–347, June 2002.

[67] T.R. Payne, M. Paolucci, and K. Sycara. Advertising and matching daml-s service descriptions. In *Proceedings of the international Semantic Web Working Symposium (SWWS)*, aug 2001.

[68] R. Pottinger and A. Halevy. Minicon : A scalable algorithm for answering queries using views. *The VLDB Journal*, 10(2-3) :182–198, 2001.

[69] R. Pottinger and A.Y. Levy. A scalable algorithm for answering queries using views. In *Proceedings of the 26th International Conference on Very Large Data Bases*, pages 484–495. Morgan Kaufmann Publishers Inc., 2000.

[70] C. Rey. D2CP et computeBCov. un prototype et un algorithme pour la découverte de services web dans le contexte du web sémantique. In J. Le Maître, editor, *Systèmes D'information Avancés*, volume 8 of *RSTI Série ISI*, pages 83 – 112. Hermes - Lavoisier, 2003.

[71] C. Rey, F. Toumani, M-S. Hacid, and A. Léger. An algorithm and a prototype for the dynamic discovery of e-services. Report, LIMOS, Clemont-Ferrand, France, 2003. see http ://www.isima.fr/∼rey/.

[72] M.-C. Rousset. Backward reasoning in aboxes for query answering. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Proceedings of the 1999 International Workshop on Description Logics (DL1999)*, 1999.

[73] K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan. Automated discovery, interaction and composition of semantic web services. *Journal of Web Semantics*, 1(1), 2003.

[74] G. Teege. Making the difference : A subtraction operation for description logics. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, pages 540–550, May 1994.

[75] D. Trastour, C. Bartolini, and C. Preist. Semantic web services : Semantic web support for the business-to-business e-commerce lifecycle. In *Proceedings of the eleventh international conference on World Wide Web*, pages 89–98. ACM Press, 2002.

[76] J.D. Ullman. Information integration using logical views. *Theoretical Computer Science*, 239(2) :189–210, 2000.

[77] Data Engineering Bulletin : Special Issue on Infrastructure for Advanced E-Services. 24(1), IEEE Computer Society, 2001.

[78] C. Wyss, C. Giannella, and E. Robertson. Fastfds : A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances. In Y. Kambayashi, W. Winiwarter, and M. Arikawa, editors, *Proceedings of the Data Warehousing and Knowledge Discovery, Third International Conference, DaWaK 2001, Munich, Germany, September 5-7*, volume 2114 of *Lecture Notes in Computer Science*. Springer, 2001.

Annexe A

Démonstrations

A.1 Résultats préliminaires pour les langages ayant la propriété de subsomption structurelle

Nous commençons par rappeler les constructeurs définissant le langage $\mathcal{L}_1$, puis nous donnons quelques résultats, concernant les langages ayant la propriété de subsomption structurelle, nécessaires aux démonstrations du lemme 3.1.1 page 30.

A.1.1 Le langage $\mathcal{L}_1$

On rappelle que le langage $\mathcal{L}_1$ est défini par les constructeurs suivants :
- $\sqcap, \sqcup, \top, \bot, (\geq n\ R), (\exists R.C)$ et $(\exists f.C)$ pour les descriptions de concepts (avec C un concept, R un rôle et f un rôle fonctionnel),
- $\bot, \circ, |$ pour les descriptions de rôles,
- $\bot, \circ$ pour les rôles fonctionnels.

Les rôles fonctionnels sont des relations binaires fonctionnelles sur l'ensemble des individus du domaine. On les appelle aussi propriétés ("features" en anglais), ou attributs. Ainsi, pour f un rôle fonctionnel et pour une interprétation $\mathcal{I}= (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$, on a :
- $f^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})$ et
- $\forall a, b, c,\ (a, b) \in f^{\mathcal{I}}$ et $(a, c) \in f^{\mathcal{I}} \rightarrow b = c$.

Pour une interprétation $\mathcal{I}= (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$, la sémantique des constructeurs de rôles est la suivante, pour R et S des rôles (fonctionnels ou non) et C un concept :
- $\bot^{\mathcal{I}} = \emptyset \subseteq (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})$
- $(R \circ S)^{\mathcal{I}} = R^{\mathcal{I}} \circ S^{\mathcal{I}} = \{(a, c) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \exists b : (a, b) \in R^{\mathcal{I}}$ et $(b, c) \in S^{\mathcal{I}}\}$
- $(R|_C)^{\mathcal{I}} = R^{\mathcal{I}} \cap \Delta^{\mathcal{I}} \times C^{\mathcal{I}}$

A.1.2 Lemmes préliminaires

Dans cette annexe, nous donnons quelques résultats utiles concernant les logiques de description ayant la propriété de subsomption structurelle. Les lemmes A.1.1 et A.1.2 sont utilisés pour démontrer le lemme A.1.3 page 140. Ce dernier lemme donne une condition nécessaire et suffisante pour qu'une clause d'une RCF d'une description de concept Q ne soit pas dans une RCF du lcs de Q avec la conjonction de plusieurs descriptions S_i. Ce lemme est nécessaire à la démonstration du lemme 3.1.1 page 30 dans le chapitre 3.

Le lemme A.1.1 ci-dessous dit que les clauses d'une RCF de $A \sqcap B$ sont les plus petites clauses (par rapport à la subsomption et modulo l'équivalence) de l'ensemble constitué des clauses d'une RCF de A et des clauses d'une RCF de B. Ce lemme est utilisé pour démontrer le lemme A.1.2.

Lemme A.1.1 (Caractérisation des RCF des conjonctions)
Soient A et B deux descriptions d'un langage ayant la propriété de subsomption structurelle et donnés par les RCF $A = \{A_1, A_2, .., A_n\}$ et $B = \{B_1, B_2, .., B_m\}$. La RCF de $A \sqcap B$ est donnée par l'ensemble :

$$C = \{A_i \in_{\equiv} A \mid \forall B_j \in_{\equiv} B,\ B_j \not\sqsubseteq A_i\} \cup \{B_j \in_{\equiv} B \mid \forall A_i \in_{\equiv} A,\ A_i \not\sqsubseteq B_j\}$$

Démonstration
Commençons par remarquer que $A \sqcap B \equiv \bigsqcap_{C_k \in_{\equiv} C} C_k$. La question qui reste est donc de savoir si C est une RCF, c'est-à-dire si on a : $\forall C_k \in_{\equiv} C, C \setminus \{C_k\} \not\sqsubseteq C_k$. Or il est facile de voir qu'on

a bien cette propriété. En effet la propriété de subsomption structurelle et la définition de C montre qu'il n'existe pas de clause $C_k \in_{\equiv} C$ telle que $C \setminus \{C_k\} \sqsubseteq C_k$. □

En supposant que le lcs des deux descriptions de concepts en RCF existe, le lemme A.1.2 ci-dessous dit qu'une clause appartient à une RCF du lcs de A et de B si et seulement si c'est le lcs d'une clause de A et d'une clause de B.

Lemme A.1.2 (Caractérisation des clauses du lcs)
Soient C et D deux descriptions de concepts d'un langage à subsomption structurelle données par leur RCF $C = \{C_j\}$ et $D = \{D_k\}$. Soit L une clause de ce langage. En supposant que le lcs de C et D existe de même que le lcs de tous les couples formés d'une clause de C et d'une clause de D, on a :

$$L \in_{\equiv} lcs(C, D) \Leftrightarrow \quad \exists (C_{j_0}, D_{k_0}) \in_{\equiv} C \times D \mid L = lcs(C_{j_0}, D_{k_0})$$

Démonstration
Pour démontrer le lemme, nous démontrons les deux implications de l'équivalence suivante :

$$\begin{aligned} L \in_{\equiv} lcs(C, D) \Leftrightarrow \quad & \exists C_{j_0} \in_{\equiv} C \mid C_{j_0} \sqsubseteq L \text{ et} \\ & \exists D_{k_0} \in_{\equiv} D \mid D_{k_0} \sqsubseteq L \text{ et} \\ & \forall \text{ clause } L', (\exists C'_j \in_{\equiv} C \mid C'_j \sqsubseteq L' \text{ et } \exists D'_k \in_{\equiv} D \mid D'_k \sqsubseteq L') \Rightarrow L' \not\equiv L \end{aligned}$$

Une fois ces deux implications démontrées, il est clair qu'une clause qui est une plus petite clause (par rapport à la subsomption et modulo l'équivalence) qui subsume une clause d'une RCF de C et une clause d'une RCF de D est le lcs des deux clauses de C et D (qui existe par hypothèse). Ce lemme est utilisé dans la démonstration du lemme A.1.3.
Implication directe $\Rightarrow$:
$L \in_{\equiv} lcs(C, D)$ donc $lcs(C, D) \sqsubseteq L$, et comme $C \sqsubseteq lcs(C, D)$, on a $C \sqsubseteq L$. d'après la propriété de subsomption structurelle, on déduit que $\exists C_{j_0} \in_{\equiv} C \mid C_{j_0} \sqsubseteq L$. On déduit de même que $\exists D_{k_0} \in_{\equiv} D \mid D_{k_0} \sqsubseteq L$.
Soit L' une clause telle que $\exists C'_j \in_{\equiv} C \mid C'_j \sqsubseteq L'$ et $\exists D'_k \in_{\equiv} D \mid D'_k \sqsubseteq L'$. Supposons que $L' \sqsubset L$.
– Premier cas : $L' \in_{\equiv} lcs(C, D)$ (sous-cas 1) ou $\exists L'' \in_{\equiv} lcs(C, D) \mid L'' \sqsubset L'$ (sous-cas 2). Puisque $L' \sqsubset L$, on a alors clairement (dans les deux sous-cas) $(lcs(C, D) \setminus L) \sqsubseteq L$ ce qui contredit le fait que L est une clause d'une RCF de $lcs(C, D)$.
– Second cas : $L' \notin_{\equiv} lcs(C, D)$ et $\forall L'' \in_{\equiv} lcs(C, D),\ L'' \not\sqsubset L'$. Alors si on remplace L par L' dans une RCF de $lcs(C, D)$ on obtient une description qui subsume toujours C et D (puisque L' subsume une clause de C et de D) mais qui est strictement subsumée par $lcs(C, D)$, ce qui contredit la définition du lcs.
Donc on a forcément $L' \not\sqsubseteq L$.
Implication réciproque $\Leftarrow$: Montrons que la contraposée est vérifiée. Supposons que $L \notin_{\equiv} lcs(C, D)$.
– Cas 1 : $\forall C_j \in_{\equiv} C,\ C_j \not\sqsubseteq L$ ou $\forall D_k \in_{\equiv} D,\ D_k \not\sqsubseteq L$.
– Cas 2 : $\exists C_{j_0} \in_{\equiv} C \mid C_{j_0} \sqsubseteq L$ et $\exists D_{k_0} \in_{\equiv} D \mid D_{k_0} \sqsubseteq L$. Dans ce cas, d'après la propriété de subsomption structurelle, on a $C \sqsubseteq L$ et $D \sqsubseteq L$. D'où, d'après la définition du lcs : $lcs(C, D) \sqsubseteq L$. D'après la propriété de subsomption structurelle, on en déduit que $\exists L' \in_{\equiv} lcs(C, D) \mid L' \sqsubseteq L$, avec $L' \not\equiv L$ car par hypothèse $L \notin_{\equiv} lcs(C, D)$. Donc $L' \sqsubset L$, et, comme $L' \in_{\equiv} lcs(C, D)$ on a aussi $\exists C'_j \in_{\equiv} C \mid C'_j \sqsubseteq L'$ et $\exists D'_k \in_{\equiv} D' \mid D'_k \sqsubseteq L'$.
La contraposée est démontrée. □

Outre le fait que ce lemme est utilisé dans la démonstration du lemme A.1.3, il nous dit que si le lcs de deux clauses existe, alors le lcs de deux descriptions existe. Cette considération est utile dans la discussion sur l'existence du lcs pour les langages ayant la propriété de subsomption structurelle à la fin de la section 1.3.1 page 17.

Lemme A.1.3
Soit Q, S_1, ..., S_n $n+1$ descriptions de concepts d'un langage ayant la propriété de subsomption structurelle, données par des RCF. Soit $E \equiv \sqcap_{k=1}^{n} S_k$ (on envisagera E soit comme conjonction soit comme ensemble des S_k). Soit $A_j \in_{\equiv} Q$. On a :

$$\forall k \in \{1, .., n\},\ A_j \notin_{\equiv} lcs(Q, S_k) \Leftrightarrow A_j \notin_{\equiv} lcs(Q, E)$$

Démonstration
D'après le lemme A.1.2, avec $A_j \in_{\equiv} Q$, on a : $A_j \in_{\equiv} lcs(Q, S_k) \Leftrightarrow \exists C_k \in_{\equiv} S_k,\ C_k \sqsubseteq A_j$. En prenant la négation de chaque membre de l'équivalence, on obtient :

$$A_j \notin_{\equiv} lcs(Q, S_k) \Leftrightarrow \forall C_k \in_{\equiv} S_k,\ C_k \not\sqsubseteq A_j$$

On a donc, pour $E \equiv \sqcap_{k=1}^{n} S_k$:

$$\begin{array}{rcl} \forall S_k, A_j \notin_{\equiv} lcs(Q, S_k) & \Leftrightarrow & \forall S_k, \forall C_k \in_{\equiv} S_k,\ C_k \not\sqsubseteq A_j \\ & \overset{lemme\ A.1.1}{\Leftrightarrow} & \forall C_E \in_{\equiv} E,\ C_E \not\sqsubseteq A_j \\ & \Leftrightarrow & A_j \notin_{\equiv} lcs(Q, E) \end{array}$$

□

A.2 Différence sémantique et syntaxique dans $\mathcal{ALN}$

Dans cette section, après le rappel de la notion de réduction d'une description, nous comparons la différence sémantique avec la différence syntaxique que l'on définit à l'occasion pour $\mathcal{ALN}$ de la même façon qu'elle l'est pour $\mathcal{ALE}$ et $\mathcal{ALC}$ (voir la section 1.3.3).

A.2.1 Réduction de la description normalisée

[50] montre que $\widehat{C}$ est une description "réduite", c'est-à-dire qu'elle est minimale par rapport à l'ordre $\preceq_d$, ce qui signifie intuitivement que $\widehat{C}$ ne contient pas de redondance d'information et que la taille de $\widehat{C}$ est minimale.

Ce résultat nous permet dans la section suivante de positionner l'opérateur de différence sémantique par rapport à l'opérateur de différence syntaxique défini dans [21] (voir le lemme A.2.4 de la section A.2.2).

Nous commençons par rappeler l'ordre $\preceq_d$ et la notion de sous-description sous-jacente pour $\mathcal{ALN}$.

Définition A.2.1 (Sous-description d'une description $\mathcal{ALN}$ [8, 50])
Soit C une $\mathcal{ALN}$-description de concept.
$\widetilde{C}$ est une sous-description de C ($\widetilde{C} \preceq_d C$) si et seulement si :

1. *$\widetilde{C} = \bot$; ou*
2. *$\widetilde{C}$ est obtenu à partir de C en enlevant le concept top, certains noms de concepts (ou négations de noms de concepts), des restrictions numériques ou des quantifications universelles au plus haut niveau de C, et pour toutes les quantifications universelles $\forall R_i.E$, on remplace E par une sous-description de E.*

$\widetilde{C}$ est une sous-description stricte de C ($\widetilde{C} \prec_d C$) si et seulement si $\widetilde{C} \preceq_d C$ et $\widetilde{C} \neq C$ (où $\widetilde{C} \neq C$ signifie que $\widetilde{C}$ et C ne s'écrivent pas de la même façon).

Définition A.2.2 (Description $\mathcal{ALN}$ réduite [8, 50])
Une $\mathcal{ALN}$-description de concept C est dite réduite si et seulement si il n'existe aucune sous-description stricte de C qui est équivalente à C (i.e. C est minimale par rapport à $\preceq_d$, ou "d-minimale").

Nous donnons maintenant le lemme qui fait le lien entre $\widehat{C}$ et C.

Lemme A.2.1 (Réduction d'une $\mathcal{ALN}$-description normalisée [50])
Soit une $\mathcal{ALN}$-description de concept C. $\widehat{C}$ est une $\mathcal{ALN}$-description réduite telle que $\widehat{C} \preceq_d C$[35].

Nous rappelons maintenant la définition de la taille d'une description et la propriété de minimalité en taille des descriptions réduites.

Définition A.2.3 (Taille d'une description $\mathcal{ALN}$ [50])
La taille $|.|$ d'une $\mathcal{ALN}$-description de concept est définie récursivement de la manière suivante :

- $|\top| := |\bot| := |A| := |\neg A| := 1$;
- $|(\geq n\ R)| := |(\leq n\ R)| := 2 + \lceil log(n+1) \rceil$ *(encodage binaire de n)* ;
- $|C \sqcap D| := |C| + |D|$;
- $|\forall R.C| := 1 + |C|$

[35] En fait ce résultat n'est pas donné sous la forme d'un lemme dans [50], mais il est quand même énoncé, après le corollaire 6.1.6.

Lemme A.2.2 (Minimalité des descriptions réduites [50])
Si E est une $\mathcal{ALN}$-description réduite et que la règle $(\forall R.E) \sqcap (\forall R.F) \xrightarrow{\equiv} (\forall R.(E \sqcap F))$ ne peut pas s'appliquer à E, alors on a pour toute $\mathcal{ALN}$-description F :

$$F \equiv E \Rightarrow |E| \leq |F|$$

On en déduit facilement le lemme suivant.

Lemme A.2.3
Soit une $\mathcal{ALN}$-description C. $\widehat{C}$ est de taille minimale.

Ce résultat implique, dans la section suivante, la minimalité en taille des solutions de la différence sémantique calculées d'après le théorème 4.2.2 page 52.

A.2.2 Comparaison avec la différence syntaxique

Nous situons maintenant la différence sémantique par rapport à la différence syntaxique définie dans [21] consistant, à partir d'une description $\mathcal{ALC}$ C et d'une description $\mathcal{ALE}$ D, à trouver la description $\mathcal{ALE}$ E telle que $E \sqcap D \equiv D \sqcap C$ et E minimale par rapport à sa taille.

Le lemme suivant découle directement du lemme A.2.3et montre que les descriptions C obtenues en résultat d'une différence $B - A$ sont telles que $\widehat{C}$ est minimale en taille. C'est le premier lien que l'on peut faire avec la différence syntaxique qui aboutit elle-même à des descriptions minimales en taille.

Lemme A.2.4
Soient deux $\mathcal{ALN}$-descriptions A et B et telles que $B \sqsubseteq A$. Les $\mathcal{ALN}$-descriptions C résultats de la différence $B - A$ construites comme indiqué dans le théorème 4.2.2 sont telles que chaque $\widehat{C}$ est minimale en taille.

Le deuxième lien, plus fort, nécessite de comparer les deux différences pour un même langage. Comme $\mathcal{ALN}$ est notre langage d'étude, on rappelle la définition de la différence syntaxique en l'appliquant à $\mathcal{ALN}$, puis on montre comment la calculer. On peut alors se rendre compte que, pour $\mathcal{ALN}$, les deux différences aboutissent aux mêmes résultats, sauf en ce qui concerne le traitement de l'inconsistance dans deux cas particuliers (déjà présentés dans la section 1.3.4).

Définition A.2.4 (Différence syntaxique dans $\mathcal{ALN}$ d'après [50, 21, 23])
Soit C et D deux $\mathcal{ALN}$-descriptions de concept. La différence (syntaxique) $B - A$ de B et A est définie comme une $\mathcal{ALN}$-description de concept C minimale par rapport à $\preceq_d$ telle que $C \sqcap A \equiv B \sqcap A$.

Voyons, à l'instar de la différence sémantique au théorème 4.2.2, comment on calcule la différence syntaxique pour $\mathcal{ALN}$.

Théorème A.2.1 (Calcul de la différence syntaxique dans $\mathcal{ALN}$)
Soient deux $\mathcal{ALN}$-descriptions A et B. La description C résultat de la différence syntaxique $B - A$ est construite ainsi :
à partir de $\widehat{\mathcal{G}}_C^{\#} := \emptyset$, les seules clauses ajoutées à $\widehat{\mathcal{G}}_C^{\#}$ sont les suivantes :

$\forall c_{AB} \in \widehat{\mathcal{G}}_{B \sqcap A}^{\#}$

- *Cas des inconsistances : si $c_{AB} = \forall R_1 ... R_n - 1. \leq 0\ R_n$ et $c_{AB} \notin \widehat{\mathcal{G}}_A^{\#}$, avec $n \geq 0$ (si $n = 0$ alors $c_{AB} = \bot$), alors c_{AB} est ajoutée à $\widehat{\mathcal{G}}_C^{\#}$*

- *Cas général : sinon si* $c_{AB} \notin \widehat{\mathcal{G}}^{\#}_{A}$, *alors* c_{AB} *est ajoutée à* $\widehat{\mathcal{G}}^{\#}_{C}$.

Démonstration
L'argument principal permettant de prouver la justesse et de la complétude de ce théorème est le fait que les clauses d'une intersection normalisée $A \sqcap B$ sont soit des clauses de A, soit des clauses de B, soit des clauses issues d'inconsistances (explicites ou implicites) entre des clauses de A et B.

La démonstration précise est donnée maintenant.
Nous commençons cette démonstration par le résultat suivant, déjà énoncé dans la section 4.3.2 page 63, qui dit qu'une clause de $A \sqcap B$ (après normalisation) provient soit de A (après normalisation), soit de B (après normalisation), soit d'une inconsistance implicite entre des clauses de A et B, ceci découlant directement des règles de normalisation (voir les règles de normalisation de la section 4.2.1 page 47).

Lemme A.2.5
Soient A et B deux descriptions $\mathcal{ALN}$. Pour chaque clause $c_{AB} \in \widehat{\mathcal{G}}^{\#}_{A\sqcap B}$, on a :
Cas 1 : soit $c_{AB} \in \widehat{\mathcal{G}}^{\#}_{A}$
Cas 2 : soit $c_{AB} \in \widehat{\mathcal{G}}^{\#}_{B}$ et $c_{AB} \notin \widehat{\mathcal{G}}^{\#}_{A}$, et alors :
Cas 2a : soit $\exists c_A \in \widehat{\mathcal{G}}^{\#}_{A} \mid c_{AB} \sqsubset c_A$ ($c_{AB} \neq c_A$ car $c_{AB} \notin \widehat{\mathcal{G}}^{\#}_{A}$)
Cas 2b : soit $\forall c_A \in \widehat{\mathcal{G}}^{\#}_{A}, c_{AB} \not\sqsubseteq c_A$ ce qui signifie, puisque $c_{AB} \in \widehat{\mathcal{G}}^{\#}_{A\sqcap B}$, que $\forall c_A \in \widehat{\mathcal{G}}^{\#}_{A}, c_A \not\sqsubseteq c_{AB}$ et qu'il n'existe pas d'inconsistance (explicite ou implicite) impliquant c_{AB} et des clauses de A.
Cas 3 : soit $c_{AB} \notin \widehat{\mathcal{G}}^{\#}_{A} \cup \widehat{\mathcal{G}}^{\#}_{B}$ et alors c_{AB} est obligatoirement issu d'une inconsistance (explicite ou implicite) impliquant des clauses de A et de B (dont c_{AB}).

Sur la base de ce lemme, on peut examiner comment construire la solution (puisqu'on va voir qu'il n'y en a qu'une) de la différence syntaxique $B - A$. Le fil directeur est que l'on doit obtenir l'égalité suivante :

$$\widehat{\mathcal{G}}^{\#}_{A\sqcap C} = \widehat{\mathcal{G}}^{\#}_{A\sqcap B}$$

puisque $A \sqcap C \equiv A \sqcap B$ (d'après le théorème 4.2.1 page 49 caractérisant la subsomption en termes d'ensembles de clauses normalisés).
Ainsi, en considérant que l'ensemble de clauses $\widehat{\mathcal{G}}^{\#}_{C}$ de notre solution C en construction est vide au début de la construction (i.e. $C \equiv \top$), et si on prend une clause c_{AB} de $\widehat{\mathcal{G}}^{\#}_{A\sqcap B}$, on en déduit :
Cas 1 : soit $c_{AB} \in \widehat{\mathcal{G}}^{\#}_{A}$.
Dans ce cas, on n'ajoute aucune clause à $\widehat{\mathcal{G}}^{\#}_{C}$ puisque cela entraîne une description C minimale par rapport à l'ordre $\preceq_d$ (puisqu'on ajoute aucune clause) en assurant que $c_{AB} \in \widehat{\mathcal{G}}^{\#}_{A\sqcap B}$ (puisque $c_{AB} \in \widehat{\mathcal{G}}^{\#}_{A}$).
Cas 2 : soit $c_{AB} \in \widehat{\mathcal{G}}^{\#}_{B}$ et $c_{AB} \notin \widehat{\mathcal{G}}^{\#}_{A}$, et alors :
Cas 2a : soit $\exists c_A \in \widehat{\mathcal{G}}^{\#}_{A} \mid c_{AB} \sqsubset c_A$.
Dans ce cas, il y a deux possibilités :
– soit $c_{AB} = \forall R_1 ... R_n{-}1. \leq 0\ R_n$ avec $n \geq 0$ (si $n = 0$ alors $c_{AB} = \bot$). Pour avoir $c_{AB} \in \widehat{\mathcal{G}}^{\#}_{A\sqcap C}$, on est alors obligé soit de mettre c_{AB} dans $\widehat{\mathcal{G}}^{\#}_{C}$, soit de mettre une clause impliquant une inconsistance avec c_A dont le résultat est c_{AB}. Clairement, l'alternative qui est minimale par rapport à $\preceq_d$ est de mettre c_{AB} dans $\widehat{\mathcal{G}}^{\#}_{C}$.
– soit $c_{AB} \neq \forall R_1 ... R_n{-}1. \leq 0\ R_n$ avec $n \geq 0$ et alors il faut que $c_{AB} \in \widehat{\mathcal{G}}^{\#}_{C}$ pour avoir $c_{AB} \in \widehat{\mathcal{G}}^{\#}_{C\sqcap A}$.

Cas 2b : soit $\forall c_A \in \widehat{\mathcal{G}}^{\#}_A, c_{AB} \not\sqsubseteq c_A$ ce qui signifie, puisque $c_{AB} \in \widehat{\mathcal{G}}^{\#}_{A\sqcap B}$, que $\forall c_A \in \widehat{\mathcal{G}}^{\#}_A, c_A \not\sqsubseteq c_{AB}$ et qu'il n'existe pas d'inconsistance (explicite ou implicite) impliquant c_{AB} et des clauses de A.

Dans ce cas, il est clair qu'il faut que $c_{AB} \in \widehat{\mathcal{G}}^{\#}_C$ pour avoir $c_{AB} \in \widehat{\mathcal{G}}^{\#}_{C\sqcap A}$.

Cas 3 : soit $c_{AB} \notin \widehat{\mathcal{G}}^{\#}_A \cup \widehat{\mathcal{G}}^{\#}_B$ et alors c_{AB} est obligatoirement issu d'une inconsistance (explicite ou implicite) impliquant des clauses de A et de B (dont c_{AB}).

Dans ce cas, comme dans le cas 2a, pour avoir $c_{AB} \in \widehat{\mathcal{G}}^{\#}_{C\sqcap A}$, il existe deux alternatives : soit mettre $c_{AB} \in \widehat{\mathcal{G}}^{\#}_C$, soit recréer l'inconsistance qui existe entre clauses de A et de B avec les mêmes clases de A et des clauses que l'on doit ajouter à C. On voit là-encore que l'alternative qui minimise C par rapport à $\preceq_d$ est la première.

En réécrivant et en résumant ces cas, on obtient l'énoncé du théorème. La complétude et la justesse proviennent de l'étude par cas qui les envisage tous (complétude) et dit ce qu'il est nécessaire de faire dans chaque cas (justesse).

□

La conséquence du théorème précédent est que, pour $\mathcal{ALN}$, les différences sémantique et syntaxique produisent le même résultat sauf dans les deux cas d'inconsistances suivants : quand une clause $\forall R_1...R_n{-}1. \leq 0\ R_n$ est dans B et une clause $\forall R_1...R_n.P$, avec $P \not\equiv \bot$, est dans A, ou bien quand il existe une inconsistance (explicite ou implicite) entre clauses de A et B. Dans ces deux cas, la différence syntaxique, imposant la minimalité par rapport à $\preceq_d$, implique une seule possibilité (là où la différence sémantique peut en impliquer plusieurs) qui consiste à garder $\forall R_1...R_n{-}1. \leq 0\ R_n$ ou la clause issue de l'inconsistance dans la description résultat (voir l'exemple 5, page 18, de la section 1.3.4).

A.3 Démonstration du lemme 4.2.2 page 51

Nous reprenons ici l'énoncé du théorème.

Soit C une $\mathcal{ALN}$-description de concept.

(i) $C \equiv \bot \quad \Leftrightarrow \quad \exists c \in C^{\#} \mid \forall c_{\uparrow} \in \widehat{\mathcal{G}}^{\#}_{Approx_{\uparrow}(\neg c)},\ \exists c' \in C^{\#} \mid\ (c' \sqsubseteq c_{\uparrow}$ et $prof(c') = prof(c_{\uparrow}))$

(ii) C présente une inconsistance implicite $\Leftrightarrow$ $\exists c \in C^{\#} \mid$
- $c = \forall R_1 R_2 ... R_{i-1}.c^*$, avec $i \geq 2$,
- $\forall c_{\uparrow} \in \widehat{\mathcal{G}}^{\#}_{Approx_{\uparrow}(\neg c^*)}, \exists c' \mid \forall R_1 R_2 ... R_{i-1}.c' \in C^{\#}$ et $c' \sqsubseteq c_{\uparrow}$ et $prof(c') = prof(c_{\uparrow})$
- $\forall R_1 R_2 ... R_{i-2}.(\geq p_{i-1}.R_{i-1}) \notin C^{\#}, \forall p_{i-1} \geq 1$

(i) caractérise les inconsistances explicites dans $\mathcal{ALN}$, et (ii) les inconsistances implicites. Dans le cas d'inconsistance implicite, la clause d'exclusion $c_{\bot}$ obtenue est :
$c_{\bot} \equiv c \sqcap (\sqcap_{c'} \forall R_1 R_2 ... R_{i-1}.c') \equiv \forall R_1 R_2 ... R_{i-2}.(\leq 0\ R_{i-1})$

Démonstration

Ce lemme s'appuie sur les lemmes 4.2.2 et 6.1.4 de [50] (issus de résultats de [49, 5, 4]) qui caractérisent les inconsistances explicites et implicites. En exprimant ces caractérisations en termes de clauses, on obtient le lemme suivant.

Lemme A.3.1
Soit C une $\mathcal{ALN}$-description de concept. On a :

- *C possède une inconsistance explicite ssi les conditions de la colonne de gauche du tableau ci-dessous sont vérifiées.*
- *C possède une inconsistance implicite aboutissant à la clause $c_{\bot} = \forall R_1 ... R_{i-i}.(\leq 0\ R_i)$ ssi les conditions de la colonne de droite du tableau A.1 ci-dessous sont vérifiées.*

Par rapport aux démonstrations des lemmes 4.2.2 et 6.1.4 de [50], dont le lemme précédent est la traduction en termes de clauses, la démonstration qui suit est une démonstration spécifique à cette thèse car elle utilise la notion de clause. On pourra donc au choix se référer à cette démonstration, ou aux démonstrations des lemmes originaux.

Démonstration
Nous allons d'abord démontrer la caractérisation des inconsistances explicites. Nous en découlerons la preuve des inconsistances implicites.

Considérons un ensemble $S_{\bot}$ de clauses minimal par inclusion et inconsistant par conjonction. D'après le lemme 4.3.1 appliqué à une seule description $\mathcal{ALN}$ C, on sait que $C^{\#}$ doit obligatoirement contenir un $S_{\bot}$ de ce type pour que C possède une inconsistance explicite. Dès lors, on a :

Il y a une inconsistance explicite dans C ssi	**Il y a une inconsistance implicite dans C ssi**
$\exists S_\perp \subseteq C^\#$ | $S_\perp = \{$ $\forall R_1 R_2 ... R_n.(P)$, $\forall R_1 R_2 ... R_n.(P')$, $\forall R_1 R_2 ... R_{n-1}.(\geq p_n R_n)$, ..., $\forall R_1.(\geq p_2 R_2)$, $(\geq p_1 R_1)$ $\}$	$\exists S^*_\perp \subseteq C^\#$ | $S^*_\perp = \{$ $\forall R_1 R_2 ... R_i R_{i+1} ... R_n.(P)$, $\forall R_1 R_2 ... R_i R_{i+1} ... R_n.(P')$, $\forall R_1 R_2 ... R_i R_{i+1} ... R_{n-1}.(\geq p_n R_n)$, ..., $\forall R_1 R_2 ... R_i.(\geq p_{i+1} R_{i+1})$ $\}$ et $\forall R_1 R_2 ... R_{i-1}.(\geq p_i R_i) \notin C^\#, \forall p_i \geq 1$.
avec : – $n \geq 0$ (si $n = 0$, alors $S_\perp = \{P, P'\}$), – P un nom de concept, la négation d'un nom de concept ou une restriction numérique, – P tel que : – soit $P' \equiv \neg P$ si P est un concept atomique ou une négation de concept atomique, – soit $P' \equiv (\leq i'_{n+1} R_{n+1})$ si $P \equiv (\geq i_{n+1} + 1\ R_{n+1})$ avec $0 \leq i'_{n+1} \leq i_{n+1}$, – soit $P' \equiv (\geq j'_{n+1} R_{n+1})$ si $P \equiv (\leq j_{n+1} - 1\ R_{n+1})$ avec $j'_{n+1} \geq j_{n+1} \geq 1$ – et $p_i \geq 1,\ \forall i \in \{1, .., n\}$. Dans ce cas d'inconsistance explicite, la clause $\perp$ est obtenue ainsi $\perp \equiv \sqcap_{c \in S_\perp} c$.	avec : – $1 \leq i \leq n$, – P un nom de concept, la négation d'un nom de concept ou une restriction numérique, – P tel que : – soit $P' \equiv \neg P$ si P est un concept atomique ou une négation de concept atomique, – soit $P' \equiv (\leq i'_{n+1} R_{n+1})$ si $P \equiv (\geq i_{n+1} + 1\ R_{n+1})$ avec $0 \leq i'_{n+1} \leq i_{n+1}$, – soit $P' \equiv (\geq j'_{n+1} R_{n+1})$ si $P \equiv (\leq j_{n+1} - 1\ R_{n+1})$ avec $j'_{n+1} \geq j_{n+1} \geq 1$ – et $p_j \geq 1,\ \forall j \in \{i+1, .., n\}$. Dans ce cas d'inconsistance implicite, la clause d'exclusion $c_\perp$ obtenue est $c_\perp = \sqcap_{c \in S^*_\perp} c \equiv \forall R_1 ... R_{i-1}.(\leq 0\ R_i)$.

TAB. A.1 – Conditions nécessaires et suffisantes caractérisant les inconsistances explicites et implicites dans une $\mathcal{ALN}$-description.

$S_\perp \neq \emptyset$
et $\bigsqcap_{c\in S_\perp} c \equiv \perp$
et $\forall c^* \in S_\perp, \bigsqcap_{c\in S_\perp\setminus\{c^*\}} c \not\equiv \perp$

$\Leftrightarrow$ $\exists c_{max} = \forall R_1 R_2 ... R_n.P \in S_\perp \mid$
- c_{max} est une clause de plus grande profondeur de $S_\perp$,
- P est un concept atomique, une négation de concept atomique ou une restriction numérique,
- $(\bigsqcap_{c\in S_\perp\setminus\{c_{max}\}} c) \sqcap c_{max} \equiv \perp$,
- $(\bigsqcap_{c\in S_\perp\setminus\{c_{max}\}} c) \not\equiv \perp$,
- $\forall c^* \in S_\perp \setminus \{c_{max}\}, (\bigsqcap_{c\in S_\perp\setminus\{c^*\}} c) \not\equiv \perp$

$\overset{def.1.2.2}{\Longrightarrow}$ $\exists c_{max} = \forall R_1 R_2 ... R_n.P \in S_\perp \mid$
- c_{max} est une clause de plus grande profondeur de $S_\perp$,
- P est un concept atomique, une négation de concept atomique ou une restriction numérique,
- $(\bigsqcap_{c\in S_\perp\setminus\{c_{max}\}} c) \sqsubseteq Approx_\uparrow(\neg c_{max})$,
- $(\bigsqcap_{c\in S_\perp\setminus\{c_{max}\}} c) \not\equiv \perp$,
- $\forall c^* \in S_\perp \setminus \{c_{max}\}, (\bigsqcap_{c\in S_\perp\setminus\{c^*\}} c) \not\equiv \perp$

$\overset{theo.4.2.1}{\Longleftrightarrow}$ $\exists c_{max} = \forall R_1 R_2 ... R_n.P \in S_\perp \mid$
- c_{max} est une clause de plus grande profondeur de $S_\perp$,
- P est un concept atomique, une négation de concept atomique ou une restriction numérique,
- $\forall c_\uparrow \in \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c_{max})}, \exists c' \in \widehat{\mathcal{G}}^{\#}_{(\sqcap_{c\in S_\perp\setminus\{c_{max}\}} c)} \mid c' \sqsubseteq c_\uparrow$,
- $(\bigsqcap_{c\in S_\perp\setminus\{c_{max}\}} c) \not\equiv \perp$,
- $\forall c^* \in S_\perp \setminus \{c_{max}\}, (\bigsqcap_{c\in S_\perp\setminus\{c^*\}} c) \not\equiv \perp$

A partir de la dernière équivalence et du lemme 1.2.1 page 13 de l'approximation faible, on peut dire que :
$\forall c_\uparrow \in \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c_{max})} = \{\forall R_1 R_2 ... R_n.(\neg P), \forall R_1 R_2 ... R_{n-1}.(\geq 1\ R_n), ..., \forall R_1.(\geq 1\ R_2), \geq 1\ R_1\}$,
$\exists c' \in \widehat{\mathcal{G}}^{\#}_{(\sqcap_{c\in S_\perp\setminus\{c_{max}\}} c)} \mid c' \sqsubseteq c_\uparrow$. Donc, on en déduit les clauses que $\widehat{\mathcal{G}}^{\#}_{(\sqcap_{c\in S_\perp\setminus\{c_{max}\}} c)}$ doit contenir.

- Pour avoir une clause subsumée par $\forall R_1 R_2 ... R_n.(\neg P)$, $\widehat{\mathcal{G}}^{\#}_{(\sqcap_{c\in S_\perp\setminus\{c_{max}\}} c)}$ doit contenir :
  - soit $\forall R_1 R_2 ... R_n.P'$ avec
    $P' \equiv \neg P$ si P est un concept atomique ou une négation de concept atomique
    ou bien
    $P' \equiv (\leq i'_{n+1} R_{n+1})$ si $\neg P \equiv (\leq i_{n+1} R_{n+1})$ avec $i'_{n+1} \leq i_{n+1}$
    ou bien
    $P' \equiv (\geq j'_{n+1} R_{n+1})$ si $(\neg P \equiv \geq j_{n+1} R_{n+1})$ avec $j'_{n+1} \geq j_{n+1}$,
  - soit $\forall R_1 R_2 ... R_{i-1}.(\leq 0\ R_i)$ avec $2 \leq i \leq n$,
  - soit $\leq 0\ R_1$,
  - soit $\perp$.
- Pour avoir une clause subsumée par $\forall R_1 R_2 ... R_{n-1}.(\geq 1\ R_n)$, $\widehat{\mathcal{G}}^{\#}_{(\sqcap_{c\in S_\perp\setminus\{c_{max}\}} c)}$ doit contenir :
  - soit $\forall R_1 R_2 ... R_{n-1}.(\geq p_n\ R_n)$ avec $p_n \geq 1$,
  - soit $\forall R_1 R_2 ... R_{i-1}.(\leq 0\ R_i)$ avec $2 \leq i \leq n-1$,

  - soit $\leq 0\ R_1$,
  - soit $\bot$.
- ...
- Pour avoir une clause subsumée par $\forall R_1.(\geq 1\ R_2)$, $\widehat{\mathcal{G}}^{\#}_{(\sqcap_{c\in S_\bot\setminus\{c_{max}\}}c)}$ doit contenir :
  - soit $\forall R_1.(\geq p_2\ R_2)$ avec $p_2 \geq 1$,
  - soit $\leq 0\ R_1$,
  - soit $\bot$.
- Pour avoir une clause subsumée par $\geq 1\ R_1$, $\widehat{\mathcal{G}}^{\#}_{(\sqcap_{c\in S_\bot\setminus\{c_{max}\}}c)}$ doit contenir :
  - soit $\geq p_1\ R_1$ avec $p_1 \geq 1$,
  - soit $\bot$.

Or $\bot$ ne peut pas appartenir à $\widehat{\mathcal{G}}^{\#}_{(\sqcap_{c\in S_\bot\setminus\{c_{max}\}}c)}$ car sinon, on aurait $(\sqcap_{c\in S_\bot\setminus\{c_{max}\}}c) \equiv \bot$, ce qui est faux par hypothèse. Donc $\widehat{\mathcal{G}}^{\#}_{(\sqcap_{c\in S_\bot\setminus\{c_{max}\}}c)}$ ne peut contenir que $\geq p_1\ R_1$ avec $p_1 \geq 1$.

Dès lors, $\widehat{\mathcal{G}}^{\#}_{(\sqcap_{c\in S_\bot\setminus\{c_{max}\}}c)}$ ne peut pas contenir $\leq 0\ R_1$, car sinon la dixième règle de normalisation (voir la section 4.2.1 47) pourrait s'appliquer à $\widehat{\mathcal{G}}^{\#}_{(\sqcap_{c\in S_\bot\setminus\{c_{max}\}}c)}$ contredisant le fait que cet ensemble de clauses est déjà normalisé. Donc $\widehat{\mathcal{G}}^{\#}_{(\sqcap_{c\in S_\bot\setminus\{c_{max}\}}c)}$ contient $\forall R_1.(\geq p_2\ R_2)$ avec $p_2 \geq 1$.

On réitère le même raisonnement pour les clauses de tailles supérieures et on obtient que $\widehat{\mathcal{G}}^{\#}_{(\sqcap_{c\in S_\bot\setminus\{c_{max}\}}c)}$ doit contenir les clauses suivantes :
- $\geq p_1\ R_1$ avec $p_1 \geq 1$,
- $\forall R_1.(\geq p_2\ R_2)$ avec $p_2 \geq 1$,
- ...,
- $\forall R_1R_2...R_{n-1}.(\geq p_n\ R_n)$ avec $p_n \geq 1$ et
- $\forall R_1R_2...R_n.P'$ avec les différents cas possibles vus précédemment.

Soit S' cet ensemble de clauses. On a donc $S' \subseteq \widehat{\mathcal{G}}^{\#}_{(\sqcap_{c\in S_\bot\setminus\{c_{max}\}}c)}$.

Par définition de $\widehat{\mathcal{G}}^{\#}_{(\sqcap_{c\in S_\bot\setminus\{c_{max}\}}c)}$, ses clauses proviennent de la normalisation des clauses de $S_\bot \setminus \{c_{max}\}$. Or, aucune règle de normalisation ne produit de clause du type $\forall R_1R_2...R_i.(\geq p_{i+1}R_{i+1})$ (avec $p_{i+1} \geq 1$ et $1 \geq i \geq n-1$) : si une clause de ce type est dans l'ensemble normalisé, c'est qu'elle était déjà dans l'ensemble non normalisé. Ainsi, la seule possibilité pour une clause de S' de ne pas être dans $S_\bot \setminus \{c_{max}\}$ est pour la clause $\forall R_1R_2...R_n.P'$ au cas où $P' \equiv \leq 0\ R_{n+1}$, c'est-à-dire au cas où $\forall R_1R_2...R_n.P'$ provient de l'application de la règle 2 ou 3 sur les clauses $\forall R_1R_2...R_nR_{n+1}.Q$ et $\forall R_1R_2...R_nR_{n+1}.(\neg Q)$, avec Q un concept atomique, une négation de concept atomique ou une restriction numérique. Cependant ce cas n'est pas non plus possible puisque n est la taille de c_{max} qui est de taille maximale dans $S_\bot$. Ainsi $S' \subseteq (S_\bot \setminus \{c_{max}\})$, ou encore $S' \cup \{c_{max}\} \subseteq S_\bot$.

Comme on connaît S', il est facile de vérifier les points suivants :
- $(\bigsqcap_{c'\in S'}c') \sqcap c_{max} \equiv \bot$,
- $(\bigsqcap_{c'\in S'}c') \not\equiv \bot$ et
- $\forall c'^* \in S', (\bigsqcap_{c'\in S'\setminus\{c'^*\}}c') \sqcap c_{max} \not\equiv \bot$.

On en déduit que, comme $S'\cup\{c_{max}\} \subseteq S_\bot$, $S'\cup\{c_{max}\} = S_\bot$. Au passage, on constate qu'aucune règle de normalisation ne s'applique à S', et donc que l'on a $S' = (S_\bot\setminus\{c_{max}\}) = \widehat{\mathcal{G}}^{\#}_{(\sqcap_{c\in S_\bot\setminus\{c_{max}\}}c)}$.

On a donc démontré que :

$$\begin{array}{ll} S_\bot \neq \emptyset & \Rightarrow \quad S_\bot = S' \cup \{c_{max}\} \\ \text{et } \bigsqcap_{c\in S_\bot} c \equiv \bot & \\ \text{et } \forall c^* \in S_\bot, \bigsqcap_{c\in S_\bot\setminus\{c^*\}} c \not\equiv \bot & \end{array}$$

La réciproque est directe : on vérifie à partir de la définition de S' que les propriétés du membre de gauche de l'équivalence s'appliquent bien à $S' \cup \{c_{max}\}$.

La précédente démonstration concerne la caractérisation des inconsistances explicites seulement. On peut facilement en déduire celle de la caractérisation des inconsistances implicites car une inconsistance implicite n'est autre qu'une inconsistance explicite à une certaine profondeur (au moins égale à 1) dans la description. □

Il reste maintenant à montrer l'équivalence entre le lemme précédent A.3.1 et le lemme 4.2.2 dont cette annexe est la démonstration.

On remarque que l'ensemble des clauses

$$\{\forall R_1 R_2 ... R_n.(P'), \forall R_1 R_2 ... R_{n-1}.(\geq p_n.R_n), ..., \forall R_1.(\geq p_2.R_2), (\geq p_1.R_1)\}$$

du lemme A.3.1 est très proche de l'ensemble

$$\widehat{\mathcal{G}}^{\#}_{Approx_{\uparrow}(\neg\forall R_1 R_2 ... R_n.(P))} = \{\forall R_1 R_2 ... R_n.(P'), \forall R_1 R_2 ... R_{n-1}.(\geq 1.R_n), ..., \forall R_1.(\geq 1.R_2), (\geq 1.R_1)\}$$

qui est l'ensemble des clauses de l'approximation faible de l' $\mathcal{ALEN}$-description $\neg\forall R_1 R_2 ... R_n.(P)$ (voir la section 1.2.2 page 13). En fait, chacune de ces clauses est subsumée en ayant même profondeur qu'une clause de $\widehat{\mathcal{G}}^{\#}_{Approx_{\uparrow}(\neg\forall R_1 R_2 ... R_n.(P))}$. Ainsi, nous pouvons reformuler les caractérisations, énoncées au lemme A.3.1, des inconsistances explicites et implicites de manière plus concise à l'aide de l'approximation faible.

□

A.4 Démonstration du théorème 4.2.2 page 52

Nous reprenons ici l'énoncé du théorème.

Soient deux $\mathcal{ALN}$-descriptions A et B telles que $B \sqsubseteq A$. Une $\mathcal{ALN}$-description $C \in B - A$ est construite ainsi : les seules clauses de $\widehat{\mathcal{G}}_C^{\#}$ sont construites de la manière suivante :

$\forall c_B \in \widehat{\mathcal{G}}_B^{\#}$

- Si (cas des inconsistances) :
  $c_B = \forall R_1 R_2 ... R_{n-1}.(\leq 0\ R_n)$, avec $n \geq 0$ (si $n = 0$ alors $c_B = \bot$)
  $\exists c_A \in \widehat{\mathcal{G}}_A^{\#} \mid c_A = \forall R_1 R_2 ... R_n R_{n+1} ... R_{n+m}.P$, $m \geq 0$
  alors, on ajoute à $\widehat{\mathcal{G}}_C^{\#}$ toutes les clauses c vérifiant :
  $\left(c = \forall R_1 R_2 ... R_n.c', c' \in \widehat{\mathcal{G}}^{\#}_{Approx_{\uparrow}(\neg \forall R_{n+1} ... R_{n+m}.P)}\right)$ et $(A \not\sqsubseteq c)$
- Sinon (cas général) : si $c_B \notin \widehat{\mathcal{G}}_A^{\#}$, alors on ajoute c_B à $\widehat{\mathcal{G}}_C^{\#}$.

D'après le premier •, on peut construire construire plusieurs $\widehat{\mathcal{G}}_C^{\#}$ pour un même couple (A, B).

Démonstration

I. Justesse partie 1 : on montre ici que chaque description C dont on a construit $\widehat{\mathcal{G}}_C^{\#}$ est telle que $C \sqcap A \equiv B$, i.e. $\widehat{\mathcal{G}}^{\#}_{C \sqcap A} = \widehat{\mathcal{G}}_B^{\#}$.

I.a) Montrons que $\widehat{\mathcal{G}}^{\#}_{C \sqcap A} \subseteq \widehat{\mathcal{G}}_B^{\#}$.

Soit $c_{AC} \in \widehat{\mathcal{G}}^{\#}_{C \sqcap A}$.

1$^{\text{er}}$cas : $c_{AC} = \forall R_1 R_2 ... R_{n-1}. \leq 0 R_n$, $n \geq 1$
soit $c_{AC} \in \widehat{\mathcal{G}}_A^{\#}$ Comme $B \sqsubseteq A$, alors $\exists c_B \in \widehat{\mathcal{G}}_B^{\#} \mid c_B \sqsubseteq c_{AC}$
  cas 1 $prof(c_B) < prof(c_{AC})$ (donc $c_B = \forall R_1 ... R_{m+1}. \leq 0 R_m$, avec $0 \leq m < n$). Par construction de C, on ajoute à $\widehat{\mathcal{G}}_C^{\#}$ un ensemble de clauses qui aboutit par inconsistance à $c_B \in \widehat{\mathcal{G}}^{\#}_{C \sqcap A}$. Comme $\widehat{\mathcal{G}}^{\#}_{C \sqcap A}$ est normalisé, la seule possibilité serait que $m = n$. CAS IMPOSSIBLE.
  cas 2 $prof(c_B) = prof(c_{AC})$. Dans ce cas, la seule possibilité est d'avoir $c_B = c_{AC}$ et donc $c_{AC} \in \widehat{\mathcal{G}}_B^{\#}$.
soit $c_{AC} \notin \widehat{\mathcal{G}}_A^{\#}$
  soit $c_{AC} \in \widehat{\mathcal{G}}_C^{\#}$ Dans le cas général, on a $c_{AC} \in \widehat{\mathcal{G}}_B^{\#}$. Dans le cas des inconsistances, c_{AC} participe à une inconsistance aboutissant à une clause de $A \sqcap C$ différente et qu'elle subsume. Donc on ne peut pas avoir $c_{AC} \in \widehat{\mathcal{G}}^{\#}_{C \sqcap A}$.
  soit $c_{AC} \notin \widehat{\mathcal{G}}_C^{\#}$ C_{AC} est obligatoirement le résultat d'un inconsistance. Par construction les seuls cas d'inconsistance dans $A \sqcap C$ aboutissent à des clauses de $\widehat{\mathcal{G}}_B^{\#}$. Donc $c_{AC} \in \widehat{\mathcal{G}}_B^{\#}$.

2$^{\text{e}}$cas : $c_{AC} \neq \forall R_1 R_2 ... R_{n-1}. \leq 0 R_n$, $n \geq 1$ D'après les règles de normalisation les clauses de ce type proviennent obligatoirement de A ou C.
soit $c_{AC} \in \widehat{\mathcal{G}}_C^{\#}$ Par construction, on a $c_{AC} \in \widehat{\mathcal{G}}_B^{\#}$.
soit $c_{AC} \in \widehat{\mathcal{G}}_A^{\#}$ Comme $B \sqsubseteq A$, alors $\exists c_B \in \widehat{\mathcal{G}}_B^{\#} \mid c_B \sqsubseteq c_{AC}$.
  soit $c_B = c_{AC}$ Donc $c_{AC} \in \widehat{\mathcal{G}}_B^{\#}$.
  soit $c_B \neq c_{AC}$ Alors, par construction, on aura dans $\widehat{\mathcal{G}}_C^{\#}$ une clause (c_B) qui, par normalisation, va faire disparaître c_{AC} de $\widehat{\mathcal{G}}^{\#}_{C \sqcap A}$, ce qui contredit les hypothèses. CAS IMPOSSIBLE.

On a donc montré que $\widehat{\mathcal{G}}^{\#}_{C\sqcap A} \subseteq \widehat{\mathcal{G}}^{\#}_{B}$.

I.b) Il découle facilement de la construction des $\widehat{\mathcal{G}}^{\#}_{C}$ que $\widehat{\mathcal{G}}^{\#}_{C\sqcap A} \supseteq \widehat{\mathcal{G}}^{\#}_{B}$: en effet, par construction, quel que soit $c_B \in \widehat{\mathcal{G}}^{\#}_{B}$, soit $\widehat{\mathcal{G}}^{\#}_{C}$ implique une inconsistance résultant en c_B dans $\widehat{\mathcal{G}}^{\#}_{C\sqcap A}$, soit c_B est dans $\widehat{\mathcal{G}}^{\#}_{C}$ et restera dans $\widehat{\mathcal{G}}^{\#}_{C\sqcap A}$.

Ainsi, pour chaque $\widehat{\mathcal{G}}^{\#}_{C}$ construit, on a bien $\widehat{\mathcal{G}}^{\#}_{C\sqcap A} = \widehat{\mathcal{G}}^{\#}_{B}$.

II. Justesse partie 2 : Montrons que chaque C construite est maximal par rapport à $\sqsubseteq$.
Soit C' tel que $C \sqsubset C'$. Montrons que l'on a forcément $C' \sqcap A \not\equiv B$.

$C \sqsubset C' \quad \Leftrightarrow \quad C \sqsubseteq C'$ et $C' \not\sqsubseteq C$
$\forall c' \in \widehat{\mathcal{G}}^{\#}_{C'},\ \exists c \in \widehat{\mathcal{G}}^{\#}_{C} \mid c \sqsubseteq c'$
et $\exists c \in \widehat{\mathcal{G}}^{\#}_{C} \mid \forall c' \in \widehat{\mathcal{G}}^{\#}_{C'},\ c' \not\sqsubseteq c$
on peut construire C' à partir de C en appliquant une ou plusieurs fois les transformations suivantes :
- soit en enlevant une clause c^* à C (i.e. $\widehat{\mathcal{G}}^{\#}_{C'} = \widehat{\mathcal{G}}^{\#}_{C} \setminus \{c^*\}$)
- soit en remplaçant une clause c^* de C par une clause c^{**} strictement moins spécifique (i.e. $\widehat{\mathcal{G}}^{\#}_{C'} = \widehat{\mathcal{G}}^{\#}_{C} \setminus \{c^*\} \cup \{c^{**}\}$ avec $c^* \sqsubset c^{**}$)

La condition suffisante de la dernière équivalence se démontre immédiatement et la condition nécessaire se démontre par contraposée.
Ainsi, il faut montrer que pour chaque C construite, si, quand on enlève une clause ou quand on remplace une clause par une clause strictement moins spécifique, alors on obtient C' telle que $C' \sqcap A \not\equiv B$.
Ainsi, soit C construite comme indiqué dans l'énoncé. Enlevons la clause c de $\widehat{\mathcal{G}}^{\#}_{C}$ pour obtenir $\widehat{\mathcal{G}}^{\#}_{C'}$.
- soit c participe à une inconsistance (explicite ou implicite) dans $C \sqcap A$ qui aboutit à une clause c_B de $\widehat{\mathcal{G}}^{\#}_{B}$ dans $\widehat{\mathcal{G}}^{\#}_{C\sqcap A}$. D'après le lemme A.3.1 (qui caractérise les inconsistances implicites et explicites), si on enlève c de $\widehat{\mathcal{G}}^{\#}_{C}$, alors, il manque dans $\widehat{\mathcal{G}}^{\#}_{C} \cup \widehat{\mathcal{G}}^{\#}_{A}$ une clause pour pouvoir réaliser l'inconsistance (en effet il n'existe pas dans $\widehat{\mathcal{G}}^{\#}_{A}$ de clause pouvant remplacer c dans l'inconsistance, puisque, par construction, c est telle que $A \not\sqsubseteq c$). On ne peut donc obtenir la clause c_B de $\widehat{\mathcal{G}}^{\#}_{B}$ dans $\widehat{\mathcal{G}}^{\#}_{C\sqcap A}$. Et donc on a forcément $C' \sqcap A \not\equiv B$.
- soit c est une clause de $\widehat{\mathcal{G}}^{\#}_{B}$ qui reste dans $\widehat{\mathcal{G}}^{\#}_{C\sqcap A}$. Si on enlève c on n'a à nouveau aucun moyen de l'obtenir dans $\widehat{\mathcal{G}}^{\#}_{C\sqcap A}$ puisque par construction, elle n'est pas dans $\widehat{\mathcal{G}}^{\#}_{A}$. Et donc on a forcément $C' \sqcap A \not\equiv B$.

Modifions maintenant la clause c de $\widehat{\mathcal{G}}^{\#}_{C}$ par clause c' strictement moins spécifique (i.e. $c \sqsubset c'$), pour obtenir $\widehat{\mathcal{G}}^{\#}_{C'}$.
- soit c participe à une inconsistance (explicite ou implicite) dans $C \sqcap A$ qui aboutit à une clause c_B de $\widehat{\mathcal{G}}^{\#}_{B}$ dans $\widehat{\mathcal{G}}^{\#}_{C\sqcap A}$. D'après la définition 1.2.2, qui dit que l'approximation faible est maximale par rapport à la subsomption, il est clair que remplacer c par une clause strictement subsumante empêche à nouveau de pouvoir réaliser l'inconsistance aboutissant la clause c_B de $\widehat{\mathcal{G}}^{\#}_{B}$ voulue. Donc on a $C' \sqcap A \not\equiv B$.
- soit c est une clause de $\widehat{\mathcal{G}}^{\#}_{B}$ qui reste dans $\widehat{\mathcal{G}}^{\#}_{C\sqcap A}$. On ne peut pas obtenir c dans $\widehat{\mathcal{G}}^{\#}_{C\sqcap A}$ si on l'a remplacée par c' puisque c n'est pas dans c est une clause de $\widehat{\mathcal{G}}^{\#}_{B}$ qui reste dans $\widehat{\mathcal{G}}^{\#}_{A}$. Donc

on a $C' \sqcap A \not\equiv B$.
Ainsi chaque C construit est maximal par rapport à la subsomption.

On conclut de tout cela que le théorème 4.2.2 est juste.

III. Complétude : Montrons que l'on construit **tous** les C maximaux par rapport à $\sqsubseteq$ tels que $C \sqcap A \equiv B$.

On veut trouver toutes les descriptions $\mathcal{ALN}$ C telles que :
– $C \sqcap A \equiv B$, c'est-à-dire
  – $\forall c_B \in \widehat{\mathcal{G}}_B^{\#}, c_B \in \widehat{\mathcal{G}}_{C\sqcap A}^{\#}$ (1)
  – et $\forall c_{CA} \in \widehat{\mathcal{G}}_{C\sqcap A}^{\#}, c_{CA} \in \widehat{\mathcal{G}}_B^{\#}$ (2)
– et C est maximal par rapport à la subsomption (3)
Soit $c_B \in \widehat{\mathcal{G}}_B^{\#}$.
• Premier cas : $c_B = \forall R_1...R_{n-1}. \leq 0 R_n, n \geq 0$ (si $n = 0$ alors $c_B = \bot$) et $S = \{c_A \in \widehat{\mathcal{G}}_A^{\#} \mid c_B \sqsubseteq c_A$ et $prof(c_B) < prof(c_A)\} \neq \emptyset$. On a donc $\forall c_A^* \in S \mid c_A^* = \forall R_1...R_n...R_{n+m}.P$, avec $m \geq 0$ et P un nom de concept, la négation d'un nom de concept ou une restriction numérique, c_A^* doit disparaître dans la normalisation de $A \sqcap C$ (car $c_A^* \notin \widehat{\mathcal{G}}_B^{\#}$ puisque $c_B \in \widehat{\mathcal{G}}_B^{\#}$). Il y a alors deux possibilités :
Possibilité 1 : disparaître à cause de la présence dans $\widehat{\mathcal{G}}_C^{\#}$ d'une clause différente et subsumée par c_A^*, c'est-à-dire que $c_B \in \widehat{\mathcal{G}}_C^{\#}$,
Possibilité 2 : disparaître à cause d'une inconsistance implicite aboutissant à c_B, c'est-à-dire que l'on doit avoir :

$$\exists S' \subseteq \widehat{\mathcal{G}}_C^{\#} \cup \widehat{\mathcal{G}}_A^{\#} \mid \left(\sqcap_{c' \in S'} c'\right) \sqcap c_A^* \equiv c_B$$

c'est-à-dire

$$\exists S'' \text{ ensemble de clauses } c'' \mid \left(\sqcap_{c'' \in S''} c''\right) \sqcap \forall R_{n+1}...R_{n+m}.P \equiv \bot$$

L'ensemble S'' de clauses $\mathcal{ALN}$ vérifiant la propriété précédente et dont la conjonction est maximale par rapport à la subsomption est $\widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg\forall R_{n+1}...R_{n+m}.P)}$ (car $(\sqcap_{c'' \in S''} c'') \sqcap \forall R_{n+1}...R_{n+m}.P \equiv \bot$ est équivalent $(\sqcap_{c'' \in S''} c'') \sqsubseteq \neg(\forall R_{n+1}...R_{n+m}.P)$ et, d'après la définition 1.2.2, $Approx_\uparrow(\neg\forall R_{n+1}...R_{n+m}.P)$ est la plus grande description $\mathcal{ALN}$ D telle que $D \sqsubseteq \neg(\forall R_{n+1}...R_{n+m}.P)$). Donc l'ensemble S' de clauses $\mathcal{ALN}$ vérifiant $(\sqcap_{c' \in S'} c') \sqcap c_A^* \equiv c_B$ et dont la conjonction est maximal par rapport à la subsomption est :

$$S' = \{c' = \forall R_1...R_n.c^*, c^* \in \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg\forall R_{n+1}...R_{n+m}.P)}\}$$

Comme $c_B \sqsubseteq c', \forall c' \in S'$, alors la possibilité qui maximise C par rapport à la subsomption est la possibilité 2, car il vaut mieux avoir $S' \subseteq \widehat{\mathcal{G}}_C^{\#} \cup \widehat{\mathcal{G}}_A^{\#}$ que $c_B \in \widehat{\mathcal{G}}_B^{\#}$. Comme il peut y avoir plusieurs c_A^*, il y aura éventuellement plusieurs solutions incomparables entre elles (on remarque au passage que mettre dans $\widehat{\mathcal{G}}_C^{\#}$ plusieurs S' issus de plusieurs c_A^* aboutit à une description C qui n'est pas maximale par rapport à la subsomption). Enfin, comme il peut exister des clauses de S' dans $\widehat{\mathcal{G}}_A^{\#}$ (voire des clauses strictement plus spécifiques impliquant la même inconsistance explicite ou implicite), alors il faut faire attention à ne pas les ajouter dans $\widehat{\mathcal{G}}_C^{\#}$ car alors les C construits ne sont pas maximaux par rapport à la subsomption. C'est pourquoi on a la condition $A \not\sqsubseteq c$.

• Autres cas : Il est facile de voir que, dans les autres cas, on ne peut obtenir c_B dans $\widehat{\mathcal{G}}^{\#}_{C\sqcap A}$ uniquement si $c_B \in \widehat{\mathcal{G}}^{\#}_{C}$ ou $c_B \in \widehat{\mathcal{G}}^{\#}_{A}$ ou les deux. Comme on veut maximiser C : si $c_B \in \widehat{\mathcal{G}}^{\#}_{A}$ alors $c_B \notin \widehat{\mathcal{G}}^{\#}_{C}$.

On a construit les $\widehat{\mathcal{G}}^{\#}_{C}$ des descriptions C vérifiant (1) et (3). Comme il est facile de voir qu'alors (2) est vérifié aussi, on a fini la démonstration.
En conclusion, le théorème 4.2.2 est complet. □

A.5 Démonstration du lemme 4.3.2 page 63

Nous reprenons ici l'énoncé du théorème.
Soient Q et E deux descriptions de concepts $\mathcal{ALN}$ consistantes. On a :

$$Q - lcs(Q,E) \not\equiv \{Q\} \Leftrightarrow \exists c_Q \in \widehat{\mathcal{G}}^{\#}_{Q} \mid c_Q \text{ est couverte par } E$$

Démonstration
Cette démonstration utilise un certain nombre des résultats donnés jusqu'ici.

$$\begin{array}{ll} Q - lcs(Q,E) \not\equiv \{Q\} \Leftrightarrow & Q - lcs(Q,E) \not\sqsubseteq \{Q\} \\ & \text{ou } Q - lcs(Q,E) \not\sqsupseteq \{Q\} \end{array}$$

Or on ne peut pas avoir $Q - lcs(Q,E) \not\sqsupseteq \{Q\}$, car $\forall C \in Q - lcs(Q,E)$[36], $C \sqcap lcs(Q,E) \equiv Q$, donc $Q \sqsubseteq C$ et donc $\{Q\} \sqsubseteq Q - lcs(Q,E)$. D'où :

$$\begin{array}{ll} & Q - lcs(Q,E) \not\equiv \{Q\} \Leftrightarrow Q - lcs(Q,E) \not\sqsubseteq \{Q\} \\ \overset{def\ 4.2.4}{\Leftrightarrow} & \exists C \in Q - lcs(Q,E) \mid C \not\sqsubseteq Q \\ \overset{theo\ 4.2.1}{\Leftrightarrow} & \exists C \in Q - lcs(Q,E) \mid \exists c_Q \in \widehat{\mathcal{G}}^{\#}_{Q} \mid \forall c_C \in \widehat{\mathcal{G}}^{\#}_{C},\ c_C \not\sqsubseteq c_Q \\ \overset{theo\ 4.2.2}{\Leftrightarrow} & \exists C \in Q - lcs(Q,E) \mid \exists c_Q \in \widehat{\mathcal{G}}^{\#}_{Q} \mid \forall c_C \in \widehat{\mathcal{G}}^{\#}_{C},\ c_C \not\sqsubseteq c_Q \text{ et} \\ & \text{soit } c_Q \in \widehat{\mathcal{G}}^{\#}_{lcs(Q,E)} \\ & \text{soit } c_Q \notin \widehat{\mathcal{G}}^{\#}_{lcs(Q,E)} \\ & \quad \text{Dans ce cas on a soit } c_Q \in \widehat{\mathcal{G}}^{\#}_{C}, \text{ ce qui contredit } \forall c_C \in \widehat{\mathcal{G}}^{\#}_{C},\ c_C \not\sqsubseteq c_Q, \\ & \quad \text{soit } c_Q = \forall R_1 ... R_{n-1}. \leq 0 R_n, \text{ avec } n \geq 1 \\ & \quad \text{et } \exists c_E \in \widehat{\mathcal{G}}^{\#}_{lcs(Q,E)} \mid c_E = \forall R_1 ... R_n ... R_{n+m}.P, \text{ avec } m \geq 0 \text{ et } P \text{ un} \\ & \quad \text{nom de concept atomique, la négation d'un nom de concept atomique} \\ & \quad \text{ou une restriction numérique.} \\ \overset{lemme\ 4.2.1}{\Leftrightarrow} & \exists C \in Q - lcs(Q,E) \mid \exists c_Q \in \widehat{\mathcal{G}}^{\#}_{Q} \mid \forall c_C \in \widehat{\mathcal{G}}^{\#}_{C},\ c_C \not\sqsubseteq c_Q \text{ et} \\ & \text{soit } c_Q \text{ est couverte directement par } E \\ & \text{soit } c_Q \text{ est couverte indirectement par } E \\ \overset{theo\ 4.2.2}{\Leftrightarrow} & \exists c_Q \in \widehat{\mathcal{G}}^{\#}_{Q} \mid c_Q \text{ est couverte par } E \end{array}$$

□

[36] On rappelle qu'une différence entre deux descriptions est un ensemble de une ou plusieurs descriptions.

A.6 Démonstration du lemme 4.3.3 page 65

Nous reprenons ici l'énoncé du théorème.

Soit Q une $\mathcal{ALN}$-description. Soit $\mathcal{T}$ une terminologie $\mathcal{ALN}$ avec $\mathcal{S}_\mathcal{T}$ l'ensemble des concepts S_i définis. Soient E_1 et E_2 deux couvertures de Q en utilisant $\mathcal{T}$. On a :

$Rest_{E_2}(Q) \sqsubseteq Rest_{E_1}(Q) \quad \Leftrightarrow \quad \forall c_Q \in \widehat{\mathcal{G}}^{\#}_Q,$
- si c_Q est couverte par E_2, alors c_Q est couverte par E_1, et
- si c_Q est indirectement couverte par E_1
  alors
  c_Q n'est pas directement couverte par E_2, et
  $\forall c_{E_2} \in \widehat{\mathcal{G}}^{\#}_{E_2} \mid c_{E_2}$ couvre indirectement c_Q, $E_1 \sqsubseteq c_{E_2}$

Démonstration

Commençons par récrire l'énoncé en remplaçant $E_1 \sqsubseteq c_{E_2}$ par $\exists c_{E_1} \in \widehat{\mathcal{G}}^{\#}_{E_1} \mid c_{E_1} \sqsubseteq c_{E_2}$, équivalent d'après la caractérisation de la subsomption structurelle (théorème 4.2.1 page 49).

Soit Q une $\mathcal{ALN}$-description. Soit $\mathcal{T}$ une terminologie $\mathcal{ALN}$ avec $\mathcal{S}_\mathcal{T}$ l'ensemble des concepts S_i définis. Soient E_1 et E_2 deux couvertures de Q en utilisant $\mathcal{T}$. On a :

$Rest_{E_2}(Q) \sqsubseteq Rest_{E_1}(Q) \quad \Leftrightarrow \quad \forall c_Q \in \widehat{\mathcal{G}}^{\#}_Q,$
- si c_Q est couverte par E_2, alors c_Q est couverte par E_1, et
- si c_Q est indirectement couverte par E_1
  alors
  c_Q n'est pas directement couverte par E_2, et
  $\forall c_{E_2} \in \widehat{\mathcal{G}}^{\#}_{E_2} \mid c_{E_2}$ couvre indirectement c_Q, $\exists c_{E_1} \in \widehat{\mathcal{G}}^{\#}_{E_1} \mid c_{E_1} \sqsubseteq c_{E_2}$

Condition nécessaire

Montrons que la contraposée est vérifiée, c'est-à-dire que l'on a :

$Rest_{E_2}(Q) \not\sqsubseteq Rest_{E_1}(Q) \quad \Leftarrow \quad \exists c^0_Q \in \widehat{\mathcal{G}}^{\#}_Q$ tel que
- soit (1) c^0_Q est couverte par E_2 et
  c^0_Q n'est pas couverte par E_1
- soit (2) c^0_Q est indirectement couverte par E_1 et
  $\exists c_{E_2} \in \widehat{\mathcal{G}}^{\#}_{E_2}$ tel que
  soit (2a) c_{E_2} couvre directement c^0_Q
  soit (2b) c_{E_2} couvre indirectement c^0_Q et
  $\forall c_{E_1} \in \widehat{\mathcal{G}}^{\#}_{E_1},\ c_{E_1} \not\sqsubseteq c_{E_2}$

Montrons que (1), (2a) et (2b) impliquent chacun $Rest_{E_2}(Q) \not\sqsubseteq Rest_{E_1}(Q)$.

(1) On a :
c^0_Q n'est pas couverte par E_1 implique $\forall R_1 \in Rest_{E_1}(Q),\ c^0_Q \in \widehat{\mathcal{G}}^{\#}_{R_1}$ (1')
c^0_Q est couverte par E_2 implique $\forall R_2 \in Rest_{E_2}(Q),\ c^0_Q \notin \widehat{\mathcal{G}}^{\#}_{R_2}$ (1")
et (1') et (1") impliquent $Rest_{E_2}(Q) \not\sqsubseteq Rest_{E_1}(Q)$.

(2a) On a :
c^0_Q est indirectement couverte par E_1 implique que pour tout $R_1 \in Rest_{E_1}(Q)$, on a au moins un ensemble de clauses c qui par conjonction avec une clause de E_1 génère une inconsistance implicite aboutissant à c^0_Q.

c^0_Q est directement couverte par E_2 implique que pour tout $R_2 \in Rest_{E_2}(Q)$, il n'y a aucune clause relative à c^0_Q, et donc aucune clause subsumée par les c précédents.
Ainsi on est sûr que $\exists R_2 \in Rest_{E_2}(Q) \mid \forall R_1 \in Rest_{E_1}(Q),\ R_2 \not\sqsubseteq R_1 \Leftrightarrow Rest_{E_2}(Q) \not\sqsubseteq Rest_{E_1}(Q)$.

(2b) Dans ce cas on a :
- c^0_Q est indirectement couverte par E_1 et
- $\exists c_{E_2} \in \widehat{\mathcal{G}}^{\#}_{E_2}$ tel que c_{E_2} couvre indirectement c^0_Q et $\forall c_{E_1} \in \widehat{\mathcal{G}}^{\#}_{E_1},\ c_{E_1} \not\sqsubseteq c_{E_2}$

$c_{E_1} \not\sqsubseteq c_{E_2} \Leftrightarrow$ (c_{E_1} et c_{E_2} sont incomparables par rapport à la subsomption) (2b') ou bien ($c_{E_1} \sqsupset c_{E_2}$) (2b").

(2b') On a c_{E_1} et c_{E_2} incomparables par rapport à la subsomption.

soit $prof(c_{E_1}) = prof(c_{E_2})$:

soit la suite des rôles dans la succession des quantifications universelles est différente, auquel cas il est clair que l'on a dans les rest de E_1 construits à partir de c_{E_1} au moins une clause incomparable avec les clauses du rest de E_2 construit à partir de c_{E_2} ;

soit la suite des rôles dans la succession des quantifications universelles est la même, i.e. $c_{E_1} = \forall R_1...R_n.P_1$, $c_{E_2} = \forall R_1...R_n.P_2$ et P_1 et P_2 sont incomparables ($n \geq 1$, P_1 et P_2 étant des concepts atomiques, des négations de concepts atomiques ou des restrictions numériques). Comme c_{E_1} et c_{E_2} couvrent indirectement c^0_Q, alors on aura forcément $c_1 = \forall R_1...R_n.\neg P_1$ dans les rest de E_1 construits à partir de c_{E_1}, et $c_2 = \forall R_1...R_n.\neg P_2$ dans les rest de E_2 construits à partir de c_{E_2}, avec c_1 et c_2 qui sont incomparables.

soit $prof(c_{E_1}) > prof(c_{E_2})$: comme précédemment, il est facile de voir (par une étude de tous les cas possibles) que l'on a dans le rest de E_1 construit à partir de c_{E_1} au moins une clause incomparable avec les clauses des rest de E_2 construits à partir de c_{E_2}. On étudiera tout de même le cas particulier suivant :
$c_{E_1} = \forall R_1...R_n...R_{n+p}.P$
$c_{E_2} = \forall R_1...R_n.(\geq 1\ R_{n+1})$
(avec $c^0_Q = \forall R_1...R_i.(\leq 0\ R_{i+1})$ avec $i < n$)
Dans ce cas, soit on retrouve la clause $c_1 = \forall R_1...R_n.(\geq 1\ R_{n+1})$ dans les rest de E_1 construits à partir de c_{E_1}, et alors c_1 est incomparable avec $c_2 = \forall R_1...R_n.(\leq 0\ R_{n+1})$ que l'on retrouve dans les rest de E_2 construits à partir de c_{E_2}, soit il existe $c = \forall R_1...R_n.(\geq p\ R_{n+1})$ avec $p \geq 1$ telle que c est une clause de E_1 (ce qui implique que les rest de E_1 construits à partir de c_{E_1} ne contiennent pas la clause c précédente, voir la caractérisation de la différence, théorème 4.2.2 page 52). Or c ne peut être une clause de E_1 puisque l'on est dans le cas où $\forall c_{E_1} \in \widehat{\mathcal{G}}^{\#}_{E_1},\ c_{E_1} \not\sqsubseteq c_{E_2}$, et on a clairement $c \sqsubseteq c_{E_2}$.

soit $prof(c_{E_1}) < prof(c_{E_2})$: comme précédemment nous n'étudierons ici qu'un seul cas pouvant sembler impliquer un rest de E_1 construit à partir de c_{E_1} subsumé par un rest de E_2 construit à partir de c_{E_2} :
$c_{E_1} = \forall R_1...R_{n-1}.(\leq 0\ R_n)$
$c_{E_2} = \forall R_1...R_n...R_{n+p}.P'$, avec $p \geq 0$
Dans ce cas, toute clause issue de c_{E_1} et présente dans les rest de E_1 est subsumée par une clause issue de c_{E_2} des rest de E_2. Cependant cette configuration ne respecte pas les hypothèses courantes, puisque c_{E_1} et c_{E_2} sont comparables. Dans tous les autres cas, on a dans le rest de E_1 construit à partir de c_{E_1} au moins une clause incomparable avec les clauses des rest de E_2 construits à partir de c_{E_2} .

(2b") On a $c_{E_1} \sqsupset c_{E_2}$.

soit $prof(c_{E_1}) = prof(c_{E_2})$: dans les rest de E_1 générés à partir de c_{E_1}, il existe obligatoirement une clause strictement subsumée par une clause des rest de E_2 générés par

c_{E_2}.

soit $prof(c_{E_1}) < prof(c_{E_2})$: impossible

soit $prof(c_{E_1}) > prof(c_{E_2})$: c'est le cas où l'on a :

$c_{E_1} = \forall R_1 ... R_n ... R_{n+p}.P$, avec $p \geq 1$

$c_{E_2} = \forall R_1 ... R_{n-1}.(\leq 0\ R_n)$

Dans ce cas, c_{E_1} génère dans les rest de E_1 la clause $c_{E_1} = \forall R_1 ... R_n ... R_{n+p}.\neg P$ qui, comme $p > 1$, est incomparable avec toutes les clauses des rest de E_2 générées à partir de c_{E_2}.

Ainsi on voit bien que dans tous les cas on a une clause des rest de E_1 issue de c_{E_1} qui ne subsume aucune clause des rest de E_2 construits à partir de c_{E_2} (soit elles sont incomparables, soit elle sont strictement subsumées) et donc :

$(\exists R_2 \in Rest_{E_2}(Q) \mid \forall R_1 \in Rest_{E_1}(Q),\ R_2 \not\sqsubseteq R_1) \Leftrightarrow (Rest_{E_2}(Q) \not\sqsubseteq Rest_{E_1}(Q))$.

Condition suffisante

On suppose vrai le membre de gauche de l'équivalence. Soit $R_2 \in Rest_{E_2}(Q)$. Montrons qu'il existe $R_1 \in Rest_{E_1}(Q)$ tel que $R_2 \sqsubseteq R_1$.

1) Soit toute clause c_Q de $\widehat{\mathcal{G}}^{\#}_Q$ couverte par E_1 l'est directement. On a $\forall c_Q \in \widehat{\mathcal{G}}^{\#}_Q$ si c_Q est couverte par E_2 alors c_Q est couverte par E_1. Donc il existe forcément un rest $R_1 \in Rest_{E_1}(Q)$ tel que $R_2 \sqsubseteq R_1$.
2) Soit il existe (au moins) une clause c_Q de $\widehat{\mathcal{G}}^{\#}_Q$ couverte indirectement par E_1. On sait alors que chaque ensemble composé d'un élément de chaque ensemble de clauses de E_1 couvrant indirectement une clause particulière de Q engendre un rest R_1 différent de E_1. Dès lors comme on a $\forall c_{E_2} \in \widehat{\mathcal{G}}^{\#}_{E_2}$, c_{E_2} ne couvre pas directement c_Q et c_{E_2} couvre indirectement c_Q $\rightarrow \exists c_{E_1} \in \widehat{\mathcal{G}}^{\#}_{E_1} \mid c_{E_1} \sqsubseteq c_{E_2}$, alors on sait que pour R_2, lui aussi déterminé par le choix des c_{E_2} de $\widehat{\mathcal{G}}^{\#}_{E_2}$ couvrant indirectement des c_Q, il existe forcément un $R_1 \in Rest_{E_1}(Q)$ tel que $R_2 \sqsubseteq R_1$ ($R_2 \sqsubseteq R_1$ provient du fait que c_{E_2} ne couvre pas directement c_Q et que si c_{E_2} couvre indirectement c_Q, alors $\exists c_{E_1} \in \widehat{\mathcal{G}}^{\#}_{E_1} \mid c_{E_1} \sqsubseteq c_{E_2}$, ce qui engendre pour les rest R_1 et R_2 correspondants la relation de subsomption).

□

A.7 Démonstration du théorème 4.4.1 page 71

Nous reprenons ici l'énoncé du théorème.
Soit $\mathcal{S}_{\mathcal{T}_\mathcal{Q}}$ un ensemble de S_i et $\mathcal{C}_Q$ l'ensemble des clauses correspondantes (i.e. $\mathcal{C}_Q = \bigcup_{S_i \in \mathcal{S}_{\mathcal{T}_\mathcal{Q}}} \widehat{\mathcal{G}}^{\#}_{S_i}$).

On a :

$$\{E_\perp \subseteq \mathcal{S}_{\mathcal{T}_\mathcal{Q}} \mid \begin{array}{l} \bigsqcap_{S_i \in E_\perp} S_i \equiv \perp \text{ et} \\ \forall S_j \in E_\perp, \bigsqcap_{S_i \in E_\perp \setminus \{S_j\}} S_i \not\equiv \perp \end{array} \}$$
$$= \quad Min_\subseteq \left\{ X \mid X \in \bigcup_{c' \in \mathcal{C}_Q} (Tr(\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}})) \right\}$$

avec, $\forall c' \in \mathcal{C}_Q$, $\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}}$ est l'hypergraphe suivant :
- l'ensemble des sommets est nommé $\Sigma_{c'}$: chaque sommet est un S_i de $\mathcal{S}_{\mathcal{T}_\mathcal{Q}}$ (et réciproquement),
- l'ensemble des arêtes est nommé $\Gamma_{c'}$: c' est une arête et les autres arêtes sont les clauses $c_\uparrow \in \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c')}$,
- $S_i \in c_\uparrow \Leftrightarrow \exists c^* \in \widehat{\mathcal{G}}^{\#}_{S_i} \mid c^* \sqsubseteq c_\uparrow$ et $prof(c^*) = prof(c_\uparrow)$

Démonstration
Nous avons d'abord besoin du résultat suivant qui est une reformulation en termes d'ensembles de clauses minimaux du lemme 4.2.2 .
Soit $\mathcal{C}_Q$ un ensemble de clauses. Soit $S_\perp \subseteq \mathcal{C}_Q$. $S_\perp$ est un plus petit ensemble de clauses (non vide) dont la conjonction est inconsistante (plus petit selon l'inclusion) si et seulement si il existe dans $S_\perp$ une clause de plus grande profondeur c_{max} telle que :
$\forall c_\uparrow \in (\{c_{max}\} \cup \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c_{max})}),\ \exists! c^* \in S_\perp \mid c^* \sqsubseteq c_\uparrow$ et $prof(c^*) = prof(c_\uparrow)$
et $\forall c^* \in S_\perp,\ \exists! c_\uparrow \in (\{c_{max}\} \cup \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c_{max})}) \mid c^* \sqsubseteq c_\uparrow$ et $prof(c^*) = prof(c_\uparrow)$

Nous allons maintenant montrer le théorème en montrant l'implication suivante. Soit $E_\perp \subseteq \mathcal{S}_{\mathcal{T}_\mathcal{Q}}$, on a :

$$\begin{array}{ll} (a) & \bigsqcap_{S_i \in E_\perp} S_i \equiv \perp \\ \quad \text{et} & \\ (b) & \forall S_j \in E_\perp, \bigsqcap_{S_i \in E_\perp \setminus \{S_j\}} S_i \not\equiv \perp \end{array} \quad \Rightarrow \quad \exists c' \in \mathcal{C}_Q \mid E_\perp \in Tr(\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}})$$

Dans la suite on utilisera "spmpq" comme abbréviation de "est subsumée par et a même profondeur que". On a :

$$\begin{array}{lll} (a) & \Leftrightarrow & \exists S_\perp \subseteq \bigcup_{S_i \in E_\perp} \widehat{\mathcal{G}}^{\#}_{S_i} \mid \\ & & \quad S_\perp \neq \emptyset \\ & & \quad \text{et } \bigsqcap_{c \in S_\perp} c \equiv \perp \\ & & \quad \text{et } S_\perp \text{ minimal par rapport à l'inclusion} \\ & & \\ & \Leftrightarrow & \exists S_\perp \subseteq \bigcup_{S_i \in E_\perp} \widehat{\mathcal{G}}^{\#}_{S_i} \mid \exists c_{max} \in S_\perp \text{de profondeur maximale} \mid \\ & & \quad (a_1) \quad \forall c_\uparrow \in (\{c_{max}\} \cup \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c_{max})}),\ \exists! c^* \in S_\perp \mid c^* \ spmpq\ c_\uparrow \\ & & \quad (a_2) \quad \text{et } \forall c^* \in S_\perp,\ \exists! c_\uparrow \in (\{c_{max}\} \cup \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c_{max})}) \mid c^* \ spmpq\ c_\uparrow \end{array}$$

Pour le c_{max} précédent, il est facile de démontrer, par contraposition, que :

$$(b) \quad \Rightarrow \quad (b_1) \qquad \forall S_j \in E_\perp, \exists c_\uparrow \in (\{c_{max}\} \cup \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c_{max})}) \mid \exists c^* \in \widehat{\mathcal{G}}^{\#}_{S_j} \mid c^* \; spmpq \; c_\uparrow \text{ et } \forall S_i \in E_\perp \setminus \{S_j\}, \forall c^{**} \in \widehat{\mathcal{G}}^{\#}_{S_i}, c^{**} \; non \; spmpq \; c_\uparrow$$

(car si le membre de droite de cette implication était faux alors le membre de gauche aussi, c'est-à-dire que $E_\perp$ ne serait pas minimal par rapport à l'inclusion.) Ainsi en ne gardant que les résultats dont on a besoin, on a :

$$\begin{aligned}
(a) \text{ et } (b) \quad \Rightarrow \quad & \exists S_\perp \subseteq \bigcup_{S_i \in E_\perp} \widehat{\mathcal{G}}^{\#}_{S_i} \mid \exists c_{max} \in S_\perp \text{de profondeur maximale } \mid \\
& (a_1) \qquad \forall c_\uparrow \in (\{c_{max}\} \cup \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c_{max})}), \; \exists! c^* \in S_\perp \mid c^* \; spmpq \; c_\uparrow \\
& (b_1) \qquad \forall S_j \in E_\perp, \exists c_\uparrow \in (\{c_{max}\} \cup \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c_{max})}) \mid \\
& \qquad\qquad \exists c^* \in \widehat{\mathcal{G}}^{\#}_{S_j} \mid c^* \; spmpq \; c_\uparrow \\
& \qquad\qquad \text{et } \forall S_i \in E_\perp \setminus \{S_j\}, \forall c^{**} \in \widehat{\mathcal{G}}^{\#}_{S_i}, c^{**} \; non \; spmpq \; c_\uparrow
\end{aligned}$$

$$\begin{aligned}
\Rightarrow \quad & \exists c_{max} \in \mathcal{C}_Q \mid \\
& \forall c_\uparrow \in (\{c_{max}\} \cup \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c_{max})}), \; \exists S_i \in E_\perp \mid \exists c^* \in \widehat{\mathcal{G}}^{\#}_{S_i} \mid c^* \; spmpq \; c_\uparrow \\
& \forall S_j \in E_\perp, \exists c_\uparrow \in (\{c_{max}\} \cup \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c_{max})}) \mid \\
& \qquad\qquad \exists c^* \in \widehat{\mathcal{G}}^{\#}_{S_j} \mid c^* \; spmpq \; c_\uparrow \\
& \qquad\qquad \text{et } \forall S_i \in E_\perp \setminus \{S_j\}, \forall c^{**} \in \widehat{\mathcal{G}}^{\#}_{S_i}, c^{**} \; non \; spmpq \; c_\uparrow
\end{aligned}$$

$$\begin{aligned}
\overset{def. \mathcal{H}_{c_{max}, \mathcal{S}_{\mathcal{T}_\mathcal{Q}}}}{\Leftrightarrow} \quad & \exists c_{max} \in \mathcal{C}_Q \mid \\
& \forall c_\uparrow \in (\{c_{max}\} \cup \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c_{max})}), \; \exists S_i \in E_\perp \mid S_i \in c_\uparrow \\
& \forall S_j \in E_\perp, \exists c_\uparrow \in (\{c_{max}\} \cup \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c_{max})}) \mid \\
& \qquad\qquad S_j \in c_\uparrow \\
& \qquad\qquad \text{et } \forall S_i \in E_\perp \setminus \{S_j\}, S_i \notin c_\uparrow
\end{aligned}$$

$$\begin{aligned}
\Rightarrow \quad & \exists c_{max} \in \mathcal{C}_Q \mid \\
& \forall c_\uparrow \in (\{c_{max}\} \cup \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c_{max})}), E_\perp \cap c_\uparrow \neq \emptyset \\
& \forall S_j \in E_\perp, \exists c_\uparrow \in (\{c_{max}\} \cup \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c_{max})}) \mid (E_\perp \setminus \{S_j\}) \cap c_\uparrow = \emptyset
\end{aligned}$$

$$\Leftrightarrow \quad \exists c_{max} \in \mathcal{C}_Q \mid E_\perp \in Tr(\mathcal{H}_{c_{max}, \mathcal{S}_{\mathcal{T}_\mathcal{Q}}})$$

On a donc :

$$\{E_\perp \subseteq \mathcal{S}_{\mathcal{T}_\mathcal{Q}} \mid \bigsqcap_{S_i \in E_\perp} S_i \equiv \perp \text{ et } \forall S_j \in E_\perp, \bigsqcap_{S_i \in E_\perp \setminus \{S_j\}} S_i \not\equiv \perp \} \quad \subseteq \quad \bigcup_{c' \in \mathcal{C}_Q} (Tr(\mathcal{H}_{c', \mathcal{S}_{\mathcal{T}_\mathcal{Q}}}))$$

Il est clair qu'un transversal minimal d'un hypergraphe $\mathcal{H}_{c',\mathcal{S}_{\mathcal{T}_\mathcal{Q}}}$, pour $c' \in \mathcal{C}_\mathcal{Q}$ est une ensemble de S_i dont la conjonction est inconsistante.
Ainsi, en ne gardant que les ensembles minimaux parmi les transversaux minimaux calculés, on obtient bien les $E_\perp$ minimaux par rapport à l'inclusion.

□

A.8 Démonstration du lemme 4.4.1 page 73

Nous reprenons ici l'énoncé du lemme.

Soit $\mathcal{S}_\mathcal{T}$ un ensemble de S_i. Soit Inc un ensemble d'ensembles E' de S_i de $\mathcal{S}_\mathcal{T}$ (i.e. $E' \subseteq \mathcal{S}_\mathcal{T}, \forall E'$). Soit l'hypergraphe $\mathcal{H}_{Inc} = (\mathcal{S}_\mathcal{T}, Inc)$. On a :

$$S_{cons} = \{E \subseteq \mathcal{S}_\mathcal{T} \mid \big(\forall E' \in Inc, E' \not\subseteq E \text{ et } E \text{ maximal par rapport à } \subseteq\big)\} = \{\mathcal{S}_\mathcal{T} \backslash X \mid X \in Tr(\mathcal{H}_{Inc})\}$$

Démonstration

Soit $E \subseteq \mathcal{S}_\mathcal{T}$ et soit
(a) $\quad \forall E' \in Inc, E' \not\subseteq E$
(b) $\quad E$ maximal par rapport à $\subseteq$ tel que (a)

D'où :

$$\begin{array}{lll} (a) & \Leftrightarrow & \forall E' \in Inc, \exists S_i \in E' \mid S_i \notin E \\ & \Leftrightarrow & \exists X \text{ un transversal de } \mathcal{H}_{Inc} \mid X \cap E = \emptyset \\ & \Leftrightarrow & \exists X \text{ un transversal minimal de } \mathcal{H}_{Inc} \mid X \cap E = \emptyset \\ & \Leftrightarrow & \exists X \text{ un transversal minimal de } \mathcal{H}_{Inc} \mid E \subseteq \mathcal{S}_\mathcal{T} \setminus X \qquad (a') \end{array}$$

Ainsi :

$$(a) \textit{ et } (b) \quad \Leftrightarrow \quad E \text{ maximal par rapport à } \subseteq \text{ tel que } (a')$$

Il est clair que les seuls sous-ensembles de $\mathcal{S}_\mathcal{T}$ maximaux par rapport à l'inclusion et suivant la propriété $\exists X$ un transversal minimal de $\mathcal{H}_{Inc} \mid E \subseteq \mathcal{S}_\mathcal{T} \setminus X$ sont les ensembles $\mathcal{S}_\mathcal{T} \setminus X$ eux-mêmes, pour $X \in Tr(\mathcal{H}_{Inc})$. □

A.9 Démonstration du lemme 4.4.2 page 81

Nous reprenons ici l'énoncé du lemme.

Soit $\mathcal{T}$ une $\mathcal{ALN}$-terminologie et Q une $\mathcal{ALN}$-description de concept. Soient C_{dir}, C_{indir} et S_{rest} les ensembles précédemment definis. Soit E^* un élément de S_{rest}, et $C_{dir}(E^*)$ et $C_{indir}(E^*)$ les ensembles définis et construits comme indiqués dans la section 4.4.

L'ensemble $C_{egal}(E^*)$ défini comme suit :

$C_{egal}(E^*)= \{(Y, c_Q) \mid c_Q \in \widehat{\mathcal{G}}^{\#}_Q$ a)
c_Q est couverte par E^* et Y b)
$Y \subseteq E^*$ et Y minimal par rapport à l'inclusion c) et
$\{c \in \widehat{\mathcal{G}}^{\#}_{E^*} \mid c_Q$ est couverte indir. par $c\} = \{c \in \widehat{\mathcal{G}}^{\#}_Y \mid c_Q$ est couverte indir. par $c\}$ d)
$\}$

peut être construit de la manière suivante :

$C_{egal}(E^*)= \{(Y, c_Q) \mid \exists (c, H, c_Q) \in C_{dir}(E^*)$ et $H = Y\}$
$\cup \{(Y, c_Q) \mid \exists (c', H', c_Q) \in C_{indir}(E^*)$ et $Y \in Min_{\subseteq}(\times_{\widetilde{c} \in \widehat{\mathcal{G}}^{\#}_{E^*}} \{\widetilde{H} | (\widetilde{c}, \widetilde{H}, c_Q) \in C_{indir}(E^*)\})\}$

Dans l'expression précédente, après le produit cartésien, on sous-entend que l'on fusionne en un seul ensemble de S_i les $\widetilde{H}$ des tuples du produit cartésien.

Démonstration

Nous montrons d'abord l'inclusion directe $\subseteq$ puis l'inclusion réciproque $\supseteq$. Tout d'abord nommons $\mathcal{J}$ l'ensemble

$\mathcal{J} = \{(Y, c_Q) \mid \exists (c, H, c_Q) \in C_{dir}(E^*)$ et $H = Y\}$
$\cup \{(Y, c_Q) \mid \exists (c', H', c_Q) \in C_{indir}(E^*)$ et $Y \in Min_{\subseteq}(\times_{\widetilde{c} \in \widehat{\mathcal{G}}^{\#}_{E^*}} \{\widetilde{H} | (\widetilde{c}, \widetilde{H}, c_Q) \in C_{indir}(E^*)\})\}$

$C_{egal}(E^*) \subseteq \mathcal{J}$:
Soit $(Y, c_Q) \in C_{egal}(E^*)$.

- 1ercas : c_Q couverte directement par E^* :
  D'après b) et d), on déduit que c_Q est couverte directement par Y et alors, d'après c) et la définition de $C_{dir}(E^*)$, on conclut que $\exists (c, H, c_Q) \in C_{dir}(E^*)$ et $H = Y$.
- 2ecas : c_Q couverte indirectement par E^* :
  Dans ce cas, par définition de $C_{indir}(E^*)$, on sait que $\exists (c', H', c_Q) \in C_{indir}(E^*)$ et $H' \subseteq E^*$. De plus, d'après d), on a $\forall \widetilde{c} \in \widehat{\mathcal{G}}^{\#}_{E^*}$, si $\widetilde{c}$ couvre indirectement une $c_Q \in \widehat{\mathcal{G}}^{\#}_Q$ alors $\widetilde{c} \in \widehat{\mathcal{G}}^{\#}_Y$. Ainsi $\forall \widetilde{c} \in \widehat{\mathcal{G}}^{\#}_{E^*}$, si $\widetilde{c}$ couvre indirectement une $c_Q \in \widehat{\mathcal{G}}^{\#}_Q$ alors $\exists \widetilde{H} \subseteq E^*$ minimal par rapport à l'inclusion et telle que $\widetilde{c} \in \widehat{\mathcal{G}}^{\#}_{\widetilde{H}}$ et $\widetilde{H} \subseteq Y$. On en déduit, comme les $\widetilde{H}$ sont minimaux par rapport à l'inclusion et comme en plus on doit avoir Y minimal par rapport à l'inclusion, que $Y \in Min_{\subseteq}(\times_{\widetilde{c} \in \widehat{\mathcal{G}}^{\#}_{E^*}} \{\widetilde{H} \mid (\widetilde{c}, \widetilde{H}, c_Q) \in C_{indir}(E^*)\})$[37].

  Ainsi, on a montré que $C_{egal}(E^*) \subseteq \mathcal{J}$.

  $C_{egal}(E^*) \supseteq \mathcal{J}$:
  Soit $(Y, c_Q) \in \mathcal{J}$.

[37] Dans l'expression précédente, après le produit cartésien, on sous-entend que l'on fusionne en un seul ensemble de S_i les $\widetilde{H}$ des tuples du produit cartésien (chaque $\widetilde{H}$ est un ensemble de S_i) avant de ne garder que les minimaux par rapport à l'inclusion de ces ensembles fusionnés. C'est-à-dire que les tuples de $\widetilde{H}$ issus du produit cartésien sont considérés comme l'union des S_i de ces $\widetilde{H}$. Cette considération simplificatrice ne change pas le résultat.

– 1$^{\text{er}}$cas : $\exists(c, H, c_Q) \in C_{dir}(E^*)$ et $H = Y$
Par définition de C_{dir} et $C_{dir}(E^*)$, cela implique que $c_Q \in \widehat{\mathcal{G}}^{\#}_{Q}$, c_Q est couverte (directement) par H, $H \subseteq E^*$ et donc c_Q couverte (directement) par E^*, et H minimal par rapport à l'inclusion. Ainsi a), b) et c) sont vérifiées. Enfin d) est vraie car c_Q est couverte directement (ce qui implique que les ensembles dont l'égalité est à vérifier sont vides tous les deux, et sont donc bien égaux).
– 2$^{\text{e}}$cas : $\exists(c', H', c_Q) \in C_{indir}(E^*)$ (*)
et $Y \in Min_{\subseteq}(\times_{\widetilde{c}\in\widehat{\mathcal{G}}^{\#}_{E^*}} \{\widetilde{H} \mid (\widetilde{c}, \widetilde{H}, c_Q) \in C_{indir}(E^*)\})$ (**)
De (*) on déduit que $c_Q \in \widehat{\mathcal{G}}^{\#}_{Q}$ et c_Q est couverte (indirectement) par E^*.
De (**) on déduit que $Y \subseteq E^*$ et que c_Q est couverte (indirectement) par Y. De plus on déduit que $\forall \widetilde{c} \in \widehat{\mathcal{G}}^{\#}_{E^*}$ tels que $\widetilde{c}$ couvre indirectement une clause c_Q de Q, $\exists \widetilde{H} \subseteq E^*$ tel que $\widetilde{c} \in \widehat{\mathcal{G}}^{\#}_{\widetilde{H}}$ et $\widetilde{H} \subseteq Y$. Donc, $\forall \widetilde{c} \in \widehat{\mathcal{G}}^{\#}_{E^*}$, si $\widetilde{c}$ couvre indirectement une clause $c_Q \in \widehat{\mathcal{G}}^{\#}_{Q}$, alors $\widetilde{c} \in \widehat{\mathcal{G}}^{\#}_{Y}$. En effet :
  – $\widehat{\mathcal{G}}^{\#}_{Y}$ ne peut pas contenir une clause c'' plus petite que $\widetilde{c}$ par rapport à la subsomption, car sinon, comme $Y \subseteq E^*$, $c'' \in \widehat{\mathcal{G}}^{\#}_{E^*}$, ce qui est impossible puisque $\widehat{\mathcal{G}}^{\#}_{E^*}$ est normalisé et $\widetilde{c} \in \widehat{\mathcal{G}}^{\#}_{E^*}$.
  – $\widehat{\mathcal{G}}^{\#}_{Y}$ ne peut pas contenir une clause c''' en inconsistance implicite avec $\widetilde{c}$ puisque, de la même façon que précédemment, cette inconsistance implicite aurait été explicitée lors de la normalisation de E^* et donc $\widetilde{c}$ ne pourrait pas être dans $\widehat{\mathcal{G}}^{\#}_{E^*}$.

Donc on a :
$\{c \in \widehat{\mathcal{G}}^{\#}_{E^*} \mid c_Q$ est couverte indir. par $c\} \subseteq \{c \in \widehat{\mathcal{G}}^{\#}_{Y} \mid c_Q$ est couverte indir. par $c\}$
Montrons l'inclusion réciproque.
Soit $c \in \widehat{\mathcal{G}}^{\#}_{Y} \mid c_Q$ est couverte indirectement par c et $c \notin \widehat{\mathcal{G}}^{\#}_{E^*}$.
$c \notin \widehat{\mathcal{G}}^{\#}_{E^*}$ implique que $\exists \widetilde{c} \in \widehat{\mathcal{G}}^{\#}_{E^*} \mid \widetilde{c} \sqsubset c$, puisque $Y \subseteq E^*$.
Puisque c_Q est couverte (indirectement) par c alors on a deux cas : soit $\widetilde{c}$ couvre aussi indirectement c_Q, mais alors, d'après l'inclusion directe montrée précédemment c aurait dû disparaître dans la normalisation de Y au profit de $\widetilde{c}$; soit $\widetilde{c}$ couvre c_Q directement, mais alors, d'après le tout premier cas, on devrait avoir dans $\widehat{\mathcal{G}}^{\#}_{Y}$ une clause c'' couvrant directement c_Q, ce qui empêche c d'appartenir à $\widehat{\mathcal{G}}^{\#}_{Y}$.
L'inclusion réciproque est ainsi démontrée.
Comme enfin on sait que Y est minimal par rapport à l'inclusion, alors on a démontré que $C_{egal}(E^*) \supseteq \mathcal{J}$.
D'où $C_{egal}(E^*) = \mathcal{J}$.

□

A.10 Démonstration du théorème 4.4.2 page 83

Nous reprenons ici l'énoncé du théorème.

Soit $\mathcal{T}$ une $\mathcal{ALN}$-terminologie et Q une $\mathcal{ALN}$-description de concept. Soient C_{dir}, C_{indir}, S_{rest} et $C_{egal}(E^*)$ les ensembles définis précédemment, avec E^* un élément de S_{rest}.

Soit $\mathcal{H}_{E^*}$ l'hypergraphe défini ainsi :

- l'ensemble des sommets est l'ensemble des Y des couples (Y, c_Q) de $C_{egal}(E^*) : \{Y \mid \exists (Y, c_Q) \in C_{egal}(E^*)\}$,
- l'ensemble des arêtes est l'ensemble des c_Q des couples (Y, c_Q) de $C_{egal}(E^*) : \{c_Q \mid \exists (Y, c_Q) \in C_{egal}(E^*)\}$
- et les arêtes sont remplies ainsi : $Y \in c_Q \Leftrightarrow (Y, c_Q) \in C_{egal}(E^*)$.

On a :

$R_{eq}(E^*) = \{X \subseteq E^* \mid X$ est minimal par rapport à l'inclusion et $Rest_{E^*}(Q) \equiv Rest_X(Q)\}$
$= Tr(\mathcal{H}_{E^*})$

Dans l'expression précédente, on sous-entend que l'on fusionne les ensembles de S_i constituants les transversaux minimaux, et que l'on ne garde que les ensembles fusionnés minimaux par inclusion.

Démonstration

L'idée principale est que la construction de $C_{egal}(E^*)$ implique un hypergraphe $\mathcal{H}_{E^*}$ dont les transversaux minimaux sont les ensembles minimaux de S_i qui suivent les conditions du lemme 4.3.5, i.e. qui ont un rest équivalent au rest de E^*.

Nommons (*) l'hypothèse suivante $X \in Tr(\mathcal{H}_{E^*})$. On a alors :

(*) $\Leftrightarrow$ $\forall c_Q$ arête de $\mathcal{H}_{E^*}$, $X \cap c_Q \neq \emptyset$ et X min pour l'inclusion

$\Leftrightarrow$ $\forall c_Q$ arête de $\mathcal{H}_{E^*}$, $\exists Y \in X \mid Y \in c_Q$ et X min pour l'inclusion

$\Leftrightarrow$ $\forall c_Q$ arête de $\mathcal{H}_{E^*}$, $\exists Y \in X \mid (Y, c_Q) \in C_{egal}(E^*)$ et X min pour l'inclusion

et comme les Y et X sont minimaux par rapport à l'inclusion, alors on peut fusionner les éléments de X , qui jusqu'ici était un ensemble d'ensembles de S_i, et qui après fusion est l'ensemble de ces S_i. C'est pourquoi on passe de $Y \in X$ à $Y \subseteq X$:

$\Leftrightarrow$ $\forall c_Q$ arête de $\mathcal{H}_{E^*}$, $\exists Y \subseteq X \mid (Y, c_Q) \in C_{egal}(E^*)$ et X min pour l'inclusion

et d'après la définition de $C_{egal}(E^*)$ on a c_Q arête de $\mathcal{H}_{E^*} \Leftrightarrow \exists (Y, c_Q) \in C_{egal}(E^*) \Leftrightarrow c_Q$ est couverte par E^*, d'où

$\Leftrightarrow$ $\forall c_Q \in \widehat{\mathcal{G}}_Q^{\#}$, si c_Q couverte par E^* alors $\exists Y \subseteq X \mid (Y, c_Q) \in C_{egal}(E^*)$ et X min pour l'inclusion

et par définition de $C_{egal}(E^*)$:

$\Leftrightarrow$ $\forall c_Q \in \widehat{\mathcal{G}}_Q^{\#}$, si c_Q couverte par E^* alors $\exists Y \subseteq X \mid$
- c_Q couverte par Y
- $Y \subseteq E^*$ et Y minimal par rapport à l'inclusion
- $\{c \in \widehat{\mathcal{G}}_{E^*}^{\#} \mid c_Q$ est couverte indirectement par $c\} =$
$\{c \in \widehat{\mathcal{G}}_Y^{\#} \mid c_Q$ est couverte indirectement par $c\}$

et X min pour l'inclusion

$\Leftrightarrow$ $\forall c_Q \in \widehat{\mathcal{G}}^{\#}_Q$,
- si c_Q couverte directement par E^* alors $\exists Y \subseteq X$ |
  - c_Q couverte directement par Y
    (car alors $\{c \in \widehat{\mathcal{G}}^{\#}_{E^*} \mid c_Q$ est couverte indirectement par $c\} = \emptyset$)
  - $Y \subseteq E^*$ et Y minimal par rapport à l'inclusion
- si c_Q couverte indirectement par E^* alors $\exists Y \subseteq X$ |
  - c_Q couverte indirectement par Y
    (car $\{c \in \widehat{\mathcal{G}}^{\#}_{E^*} \mid c_Q$ est couverte indir. par $c\} = \{c \in \widehat{\mathcal{G}}^{\#}_{Y} \mid c_Q$ est couverte indir. par $c\}$)
  - $Y \subseteq E^*$ et Y minimal par rapport à l'inclusion

  et X min pour l'inclusion

et en considérant X comme l'ensemble des S_i des P correspondants on obtient :

$\Leftrightarrow$
- $\{c \in \widehat{\mathcal{G}}^{\#}_{E^*} \mid c_Q$ est couverte dir. par $c\} = \{c \in \widehat{\mathcal{G}}^{\#}_{X} \mid c_Q$ est couverte dir. par $c\}$
  car
  - l'inclusion $\subseteq$ est montrée à l'équivalence précédente,
  - l'inclusion $\supseteq$ est vraie car $X \subseteq E^*$ puisque X est formé de (plusieurs) Y et $Y \subseteq E^*$ et
  - les clauses des $\widehat{\mathcal{G}}^{\#}_Y$ restent dans $\widehat{\mathcal{G}}^{\#}_X$ car, par construction de $C_{indir}(E^*)$, elles sont minimales par rapport à la subsomption (dans E^* donc dans les Y et donc dans X aussi).
- $\forall c_Q \in \widehat{\mathcal{G}}^{\#}_Q$, $\{c \in \widehat{\mathcal{G}}^{\#}_{E^*} \mid c_Q$ est couverte indirectement par $c\} =$
  $\{c \in \widehat{\mathcal{G}}^{\#}_X \mid c_Q$ est couverte indirectement par $c\}$ (provient des arguments précédents.

et d'après le lemme 4.3.5

$\Leftrightarrow$ $X \subseteq E^* \mid X$ est minimal par rapport à l'inclusion et $Rest_{E^*}(Q) \equiv Rest_X(Q)$

□

A.11 Démonstration de la complexité d'$\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$ (théorème 4.5.1 page 88)

Le principe de cette démonstration est de réduire polynomialement le problème $\mathcal{BCOV}(\mathcal{T}, Q)$ pour $\mathcal{FL}_0$ (qui est NP-Difficile), en le problème $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$.

Commençons par rappeler les définitions des notions de meilleures couvertures pour les langages à subsomption structurelle et pour $\mathcal{ALN}$. Pour $\mathcal{T}$ une terminologie, Q et E deux descriptions, on a :

Si $\mathcal{T}$, Q et E sont exprimées dans un langage à subsomption structurelle, alors E est une meilleure couverture de Q selon $\mathcal{T}$ si et seulement si	Si $\mathcal{T}$, Q et E sont exprimées dans $\mathcal{ALN}$, alors E est une meilleure couverture de Q selon $\mathcal{T}$ si et seulement si
– E est une conjonction de S_i de $\mathcal{T}$ (i.e. $S_i \in \mathcal{S}_\mathcal{T}$) – $Q - lcs(Q, E) \not\equiv Q$ – $\lvert Rest_E(Q)\rvert$ est minimale – puis $\lvert Miss_E(Q)\rvert$ est minimale – puis E est minimale par rapport à l'inclusion.	– E est une conjonction de S_i de $\mathcal{T}$ (i.e. $S_i \in \mathcal{S}_\mathcal{T}$) – $Q \sqcap E \not\equiv \bot$ – $Q - lcs(Q, E) \not\equiv \{Q\}$ – $Rest_E(Q)$ est maximal par rapport à la subsomption – puis $Miss_E(Q)$ est maximal par rapport à la subsomption – puis E est minimale par rapport à l'inclusion.
avec $\lvert D\rvert$ est le nombre de clauses de la forme clausale reduite de D (avec D une description exprimée dans une logique de description à subsomption structurelle).	

Avant de montrer le lien entre ces définitions pour $\mathcal{FL}_0$, nous donnons quelques résultats utiles dans le lemme suivant.

Lemme A.11.1

1 *Soient c_1 et c_2 deux clauses $\mathcal{FL}_0$ au sens $\mathcal{ALN}$, c'est-à-dire que $c_1 = \forall R_1 R_2 ... R_n.P$ avec $n \geq 0$ et P un concept atomique (si $n = 0$ alors $c_1 = P$), et pareil pour c_2. On a : $c_1 \sqsubseteq c_2 \Leftrightarrow c_1 = c_2$.*

2 *Soient D_1 et D_2 deux descriptions $\mathcal{FL}_0$. On a : $\widehat{\mathcal{G}}^{\#}_{lcs(D_1,D_2)} = \widehat{\mathcal{G}}^{\#}_{D_1} \cap \widehat{\mathcal{G}}^{\#}_{D_2}$.*

3 *Soient D_1 et D_2 deux descriptions $\mathcal{FL}_0$. On a : $\widehat{\mathcal{G}}^{\#}_{D_1 - lcs(D_1,D_2)} = \widehat{\mathcal{G}}^{\#}_{D_1} \setminus \widehat{\mathcal{G}}^{\#}_{D_2}$.*

4 *Soient D_1 et D_2 deux descriptions $\mathcal{FL}_0$. On a : $D_2 \sqsubseteq D_1 \Rightarrow |D_1| \leq |D_2|$*[38].

Démonstration
Le résultat 1 est trivial.
Le résultat 2 est basé sur le lemme 4.2.1 et sur le résultat 1.
Le résultat 3 est basé sur le fait que la différence de Teege se traduit par une différence ensembliste au niveau des ensembles de clauses pour les langages à subsomption structurelle (voir la section 1.3.1), ainsi que sur le résultat 2.
Le résultat 4 est basé sur la caractérisation de la subsomption dans $\mathcal{ALN}$ avec les clauses (voir le théorème 4.2.1), sur le résultat 3 et sur le fait que les notions de clauses dans les langages à subsomption structurelle et dans $\mathcal{ALN}$ coïncident (voir les définitions 1.3.2 page 15 et 4.2.1 page 48) car la normalisation dans $\mathcal{ALN}$ supprime de la description les clauses qui subsument la conjonction de l'ensemble des autres (voir les règles de normalisation de la section 4.2.1 page 47), ce qui est la définition de la forme clausale réduite pour les langages à subsomption structurelle.
□

[38] On rappelle ici que la taille |.| d'une description dans un langage à subsomption structurelle est le nombre de noms de rôles et de concepts dans sa RCF. Voir la définition 3.1.3 page 28.

A partir du lemme A.11.1, on explicite le lien qui existe entre $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$ et $\mathcal{BCOV}(\mathcal{T}, Q)$ pour $\mathcal{FL}_0$, et on en déduit la complexité de $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$.

On rappelle que l'on veut montrer les points suivants (qui constituent le théorème 4.5.1) :

- La recherche des couvertures de meilleur rest dans $\mathcal{FL}_0$ est le même problème quand $\mathcal{FL}_0$ est considéré comme un langage à subsomption structurelle ou comme un sous-langage d'$\mathcal{ALN}$.
- Pour $\mathcal{FL}_0$, les solutions de $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$ (les couvertures ayant meilleurs rest et miss et minimales par rapport à l'inclusion) englobent les solutions de $\mathcal{BCOV}(\mathcal{T}, Q)$.
- $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$ est un problème NP-Difficile, pour $\mathcal{T}$ une $\mathcal{ALN}$-terminologie et Q une $\mathcal{ALN}$-description.

Démonstration

Soit $\mathcal{T}$ une $\mathcal{FL}_0$-terminologie, Q et E deux descriptions $\mathcal{FL}_0$. Montrons d'abord que E est une couverture de Q selon $\mathcal{T}$ de meilleur rest avec $\mathcal{FL}_0$ considéré comme un langage à subsomption structurelle si et seulement si E est une couverture de Q selon $\mathcal{T}$ de meilleur rest avec $\mathcal{FL}_0$ considéré comme sous-langage d'$\mathcal{ALN}$. On a :

– Dans les deux cas E est une conjonction de S_i de $\mathcal{T}$.

– On a vu précédemment (voir le paragraphe suivant la définition 3.1.1 page 28) que pour les langages à subsomption structurelle, on avait toujours $E \sqcap Q \not\equiv \bot$. Donc dans les deux cas on a $E \sqcap Q \not\equiv \bot$.

– $Q - lcs(Q, E) \not\equiv \{Q\} \Leftrightarrow Q - lcs(Q, E) \not\equiv Q$ car la définition de la subsomption entre ensemble de descriptions (voir la définition 4.2.4 page 54) est telle que $(\{A\} \equiv \{B\}) \Leftrightarrow (A \equiv B)$.
A titre purement facultatif, on peut vérifier que $Q - lcs(Q, E) \not\equiv \{Q\} \Leftrightarrow Q - lcs(Q, E) \not\equiv Q$ par les équivalences suivantes :

$$
\begin{array}{lll}
Q - lcs(Q,E) \not\equiv \{Q\} & \overset{lemme\ 4.3.2}{\Leftrightarrow} & \exists (c_Q, c_E) \in \widehat{\mathcal{G}}_Q^{\#} \times \widehat{\mathcal{G}}_E^{\#} \mid c_Q \text{ est couverte par } c_E \\
& \overset{pour\ \mathcal{FL}_0}{\Leftrightarrow} & \exists (c_Q, c_E) \in \widehat{\mathcal{G}}_Q^{\#} \times \widehat{\mathcal{G}}_E^{\#} \mid c_Q \text{ est couverte directement par } c_E \\
& \Leftrightarrow & \exists (c_Q, c_E) \in \widehat{\mathcal{G}}_Q^{\#} \times \widehat{\mathcal{G}}_E^{\#} \mid c_E \sqsubseteq c_Q \\
& \overset{resultat\ 1}{\Leftrightarrow} & \exists (c_Q, c_E) \in \widehat{\mathcal{G}}_Q^{\#} \times \widehat{\mathcal{G}}_E^{\#} \mid c_E = c_Q \\
& \Leftrightarrow & \widehat{\mathcal{G}}_Q^{\#} \cap \widehat{\mathcal{G}}_E^{\#} \neq \emptyset \\
& \Leftrightarrow & \widehat{\mathcal{G}}_Q^{\#} \setminus \widehat{\mathcal{G}}_E^{\#} \neq \widehat{\mathcal{G}}_Q^{\#} \\
& \Leftrightarrow & \widehat{\mathcal{G}}_Q^{\#} \setminus (\widehat{\mathcal{G}}_E^{\#} \cap \widehat{\mathcal{G}}_Q^{\#}) \neq \widehat{\mathcal{G}}_Q^{\#} \\
& \overset{resultat\ 2}{\Leftrightarrow} & \widehat{\mathcal{G}}_Q^{\#} \setminus \widehat{\mathcal{G}}_{lcs(Q,E)}^{\#} \neq \widehat{\mathcal{G}}_Q^{\#} \\
& \overset{resultat\ 3}{\Leftrightarrow} & \widehat{\mathcal{G}}_{Q-lcs(Q,E)}^{\#} \neq \widehat{\mathcal{G}}_Q^{\#} \\
& \overset{theo\ 4.2.1}{\Leftrightarrow} & Q - lcs(Q,E) \not\equiv Q
\end{array}
$$

– $|Rest_E(Q)|$ est minimale $\Leftrightarrow$ $Rest_E(Q)$ est maximal par rapport à la subsomption.
En effet, comme $\mathcal{FL}_0$ ne permet pas d'exprimer l'inconsistance, si des couvertures E_1 et E_2 ont des rest incomparables par rapport à la subsomption, alors on peut toujours former $E_1 \sqcap E_2$ dont le rest subsumera obligatoirement les rest de E_1 et E_2. Ainsi, le rest maximal par rapport à la subsomption existe et est unique, et ce rest maximal est celui de la conjonction de tous les S_i de $\mathcal{T}$. Or on a montré dans le lemme 3.1.1 page 30 que le rest de la conjonction de tous les S_i était de taille minimale. D'où l'équivalence.

Nous avons donc montré que les définitions des couvertures de meilleur rest sont équivalentes pour $\mathcal{FL}_0$ considéré comme sous-langage d'$\mathcal{ALN}$ ou langage à subsomption structurelle.

Le premier point du théorème est donc démontré. Montrons maintenant que les meilleures couvertures qui maximisent le miss par rapport à la subsomption contiennent les meilleures couvertures qui minimisent la taille du miss.
Supposons que E soit une couverture d'une $\mathcal{FL}_0$-description Q avec un miss de taille minimale. On a donc : $\forall E', |Miss_{E'}(Q)| \geq |Miss_E(Q)|$. Supposons de plus que le miss de E ne soit pas maximal par rapport à la subsomption. On a alors $\exists E'' \mid Miss_E(Q) \sqsubset Miss_{E''}(Q)$. D'après le résultat 4 du lemme précédent, on en déduit que $|Miss_E(Q)| > |Miss_{E''}(Q)|$, ce qui contredit la première hypothèse. On a donc démontré qu'une couverture dont la taille du miss est minimale a un miss maximal par rapport à la subsomption. Cependant la réciproque n'est pas vraie. L'exemple suivant le prouve :

$Q \equiv \forall R_1 R_2.P_1 \sqcap \forall R_1 R_3.P_2$
$S_1 \equiv \forall R_1 R_2.P_1 \sqcap \forall R_4.P_3$
$S_2 \equiv \forall R_1 R_3.P_2 \sqcap \forall R_5.P_7 \sqcap \forall R_6.P_8$
$Miss_{S_1}(Q) \equiv \forall R_4.P_3$ et $|Miss_{S_1}(Q)| = 2$
$Miss_{S_2}(Q) \equiv \forall R_5.P_7 \sqcap \forall R_6.P_8$ et $|Miss_{S_2}(Q)| = 4$
Il est clair que $Miss_{S_1}(Q)$ et $Miss_{S_2}(Q)$ sont maximaux par rapport à la subsomption car incomparables (et car la seule autre couverture possible est $S_1 \sqcap S_2$ dont le miss est subsumé par les deux miss précédents, puisque $Miss_{S_1 \sqcap S_2}(Q) \equiv \forall R_4.P_3 \sqcap \forall R_5.P_7 \sqcap \forall R_6.P_8$). Or S_2 n'a pas un miss de taille minimal.

Comme le dernier critère définissant les meilleures couvertures, qui est celui de la minimalité par rapport à l'inclusion des couvertures considérées comme ensembles de S_i, est présent dans $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$ et $\mathcal{BCOV}(\mathcal{T}, Q)$, on en conclut que résoudre $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$ est au moins aussi difficile que résoudre $\mathcal{BCOV}(\mathcal{T}, Q)$ puisque que les solutions de $\mathcal{BCOV}(\mathcal{T}, Q)$ sont inclues dans celles de $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$. Comme $\mathcal{BCOV}(\mathcal{T}, Q)$ est NP-Difficile, alors $\mathcal{ALN}$-$\mathcal{BCOV}(\mathcal{T}, Q)$ est aussi NP-Difficile. □

Annexe B

Calcul de la complexité de l'algorithme *computeALNBCov*

On rappelle que les entrées de l'algorithme sont une $\mathcal{ALN}$-terminologie $\mathcal{T}$ dont l'ensemble des concepts définis S_i est nommé $\mathcal{S}_\mathcal{T}$ (sachant que chaque S_i est donné par son ensemble de clauses normalisées $\widehat{\mathcal{G}}^{\#}_{S_i}$), ainsi qu'une $\mathcal{ALN}$-description de concept Q elle aussi donnée par son ensemble de clauses normalisées $\widehat{\mathcal{G}}^{\#}_{Q}$. On nomme $\mathcal{S}_{\mathcal{T}_\mathcal{Q}} = \mathcal{S}_\mathcal{T} \cup \{Q\}$. Pour le calcul de la complexité, on pose :

- $v = |\mathcal{S}_\mathcal{T}|$, i.e. v est le nombre de concepts définis S_i de $\mathcal{T}$,
- $p = Max(prof(c) | c \in \widehat{\mathcal{G}}^{\#}_{S_1} \cup ... \cup \widehat{\mathcal{G}}^{\#}_{S_v} \cup \widehat{\mathcal{G}}^{\#}_{Q})$, i.e. p est la profondeur maximale d'une clause présente dans un $\widehat{\mathcal{G}}^{\#}_{S_i}$ dans $\widehat{\mathcal{G}}^{\#}_{Q}$,
- $l = Max_i(|\widehat{\mathcal{G}}^{\#}_{S_i}|)$, i.e. l est le nombre de clauses maximal d'un $\widehat{\mathcal{G}}^{\#}_{S_i}$ et
- $N = |\widehat{\mathcal{G}}^{\#}_{Q}|$, i.e. N est le nombre de clauses de $\widehat{\mathcal{G}}^{\#}_{Q}$.

L'étude de complexité est organisée en quatre parties : d'abord, nous étudierons la complexité au pire du calcul des transversaux minimaux quand on utilise l'algorithme 2 page 37 (implémentant le théorème 3.2.1 des persistants), puis celle du calcul des inconsistances explicites et implicites et de la construction des ensembles C_{dir} et C_{indir}, ensuite, en fonction du nombre d'inconsistances obtenues à l'étape précédente, nous donnerons la complexité au pire du calcul des couvertures de rest maximal, et enfin nous terminerons par donner la complexité au pire du calcul des meilleures couvertures. Nous terminerons par une discussion sur les résultats de complexité obtenus.

Comme résultats de complexité au pire, on donnera des bornes supérieures du temps d'exécution des calculs considérés ainsi que des bornes supérieures sur les cardinalités des ensembles construits. Les bornes supérieures en temps ne seront pas cumulatives, c'est-à-dire que nous donnerons seulement les temps nécessaires au calcul de chaque étape de *computeALNBCov* en ne cumulant pas les temps de calcul des étapes précédentes. Par exemple, le temps de calcul de S_{couv} sera évalué en considérant que S_{cons}, C_{dir} et C_{indir} sont des donnés d'entrées de ce calcul, et non en cumulant le temps nécessaire à leur construction.

Complexité du calcul de transversaux minimaux avec les persistants

Nous faisons l'hypothèse que le pire cas pour un calcul de transversaux minimaux dans un hypergraphe est celui du produit cartésien, c'est-à-dire que toutes les arêtes sont disjointes. Il

est facile de voir que, dans ce cas, le nombre de transversaux minimaux est exponentiel en fonction du nombre d'arêtes m et du nombre maximal de sommets dans une arête, nombre borné par le nombre total n de sommets dans l'hypergraphe. Ainsi la borne supérieure est exponentielle en $\mathcal{O}(n^m)$ (voir par exemple l'hypergraphe H_1 à l'exemple 37 page 178). Cette borne supérieure concerne le nombre maximal de transversaux minimaux que l'on peut obtenir dans un hypergraphe.

Une borne supérieure du temps nécessaire au calcul des transversaux minimaux est donnée par $\mathcal{O}((m+1).n^m)$. En effet, en plus des générations de transversaux minimaux à chaque itération, dont le nombre total est majoré par n^m comme on vient de le voir, il faut, à chaque itération, calculer l'intersection de chaque transversal minimal calculé à l'itération précédente avec l'arête de l'itération courante afin de déterminer les 1-, n- et non-persistants, puis tester les éventuels cas des non-persistants générant des transversaux non minimaux à l'itération courante[39]. Le calcul des intersections entre l'arête courante et les transversaux minimaux de l'itération précédente, si l'on suppose que les ensembles de sommets étudiés sont triés, est majoré par $n*1$ à l'itération 1, $n*n$ à l'itération 2,..., $n*n^m$ à l'itération m. En majorant chaque terme par n^m on obtient $m.n^m$. Les tests d'inclusion strict entre 1-persistants privé d'un sommet de l'arête courante et les non-persistants ne coûtent rien puisque dans le cas du produit cartésien il n'y a aucun 1-persistant. Ainsi, la somme des intersections et des générations est bien majorée par $(m+1).n^m$.

Dans la suite, on majorera donc le coût de chaque calcul de transversaux minimaux ainsi :
- nombre de transversaux minimaux générés $\leq n^m$ et
- temps maximal de calcul $\mathcal{O}((m+1).n^m)$
- pour n le nombre de sommets de l'hypergraphe et m le nombre d'arêtes.

Complexité du calcul des inconsistances

Le calcul de tous les cas d'inconsistances explicites et implicites se fait par un ensemble de calculs de transversaux minimaux d'hypergraphes. Il y a un calcul de transversaux minimaux pour chaque clause c' de $\mathcal{C}_Q = \bigcup_{S_i \in \mathcal{S}_{\mathcal{T}_\mathcal{Q}}} \widehat{\mathcal{G}}^{\#}_{S_i}$, soit au pire $v*l+N$. Pour chaque clause c', un hypergraphe est créé de la manière suivante : les sommets sont les S_i et Q, soit $v+1$ sommets, et les arêtes correspondent aux clauses c' et $c_\uparrow \in \widehat{\mathcal{G}}^{\#}_{Approx_\uparrow(\neg c')}$. D'après le lemme 4.2.2 page 51, le nombre de clauses de l'approximation faible de la négation d'une clause est donné par la profondeur de cette clause +1. Ainsi, quelle que soit la clause c', l'hypergraphe aura au plus $p+1$ arêtes. Pour construire chaque arête (au plus au nombre de $p+1$ pour chaque hypergraphe), on examine toutes les clauses des S_i et de Q, au plus $vl+N$, en testant si elles sont subsumées et de même profondeur, avec un coût de p, que la clause correspondant à l'arête, soit un coût total d'au plus $(p+1)(vl+N)p$ pour chaque hypergraphe. La somme, pour tous les hypergraphes, des temps de construction et de calcul des transversaux minimaux correspondants est de $(vl+N)[(p+1)(vl+N)p+(p+2)(v+1)^{p+1}]$, ce qu'on peut majorer par $2(vl+N)(p+2)^2(v+1)^{p+1}$. D'après le paragraphe précédent, on obtiendra au plus $(vl+N)(v+1)^{p+1}$ transversaux minimaux (correspondant aux inconsistances explicites), qui auront nécessité le calcul d'au plus $(vl+N)p(v+1)^p$ non-persistants intermédiaires (correspondant aux inconsistances implicites). Ensuite, pour calculer Inc_Q, il suffit de trier tous les transversaux minimaux obtenus par rapport

[39] On rappelle que l'union d'un non persistant X et d'un sommet S de l'arête courante est un transversal non minimal si et seulement si il existe un 1-persistant X' dont l'intersection avec l'arête courante est $\{S\}$ et tel que $X' \setminus \{S\} \subset X$.

à l'inclusion. En les comparant tous 2 à 2, avec un coût de $v+1$ par test d'inclusion, on obtient une borne supérieure du temps de tri de $(vl+N)^2(v+1)^{2(p+1)}(v+1)$. Le temps total de calcul de Inc_Q est donc de $\mathcal{O}(2(vl+N)(p+2)^2(v+1)^{p+1}+(vl+N)^2(v+1)^{2(p+1)}(v+1))$. En majorant notamment $(p+2)^2$ par $(v+1)^p$, on obtient $\mathcal{O}(3(vl+N)^2(v+1)^{2(p+1)})$ ou encore $\mathcal{O}(3l^2N^2(v+1)^{2p+4})$. En résumé, on a :

- le nombre de transversaux minimaux calculés (i.e. le nombre d'inconsistances explicites), et donc $|Inc_Q|$, est $\leq (vl+N)(v+1)^{(p+1)}$,
- le nombre de non-persistants intermédiaires (i.e. le nombre d'inconsistances implicites) est $\leq (vl+N)p(v+1)^p$,
- le temps de calcul de Inc_Q est $\mathcal{O}(l^2N^2(v+1)^{2p+4})$.

Au pire Inc possède autant d'élément que Inc_Q. Calculer Inc revient à examiner tous les éléments de Inc_Q qui sont des ensembles de S_i de $\mathcal{S}_{\mathcal{T}_\mathcal{Q}}$ ayant une cardinalité $\leq v+1$, puis à les comparer deux à deux par rapport à l'inclusion (pour ne garder que les plus petits). Donc le temps maximal d'examen est $\mathcal{O}((v+1)*(vl+N)(v+1)^{p+1})$, ou $\mathcal{O}((vl+N)(v+1)^{p+2})$, ou encore $\mathcal{O}(lN(v+1)^{p+3})$. Le temps maximal de tri est $\mathcal{O}((v+1)*((vl+N)(v+1)^{p+1})^2)$, ou $\mathcal{O}(l^2N^2(v+1)^{2p+5})$. Au total, on a un temps de $\mathcal{O}(l^2N^2(v+1)^{2p+5})$. Ainsi :

- $|Inc| \leq (vl+N)(v+1)^{p+1}$ et
- le temps de calcul de Inc est $\mathcal{O}(l^2N^2(v+1)^{2p+5})$.

Calculer S_{cons} revient à chercher le complémentaire dans $\mathcal{S}_\mathcal{T}$ de chaque transversal minimal X de $\mathcal{H}_{Inc}$. $\mathcal{H}_{Inc}$ a comme sommets les S_i de $\mathcal{S}_\mathcal{T}$ (au nombre de v) et comme arêtes les éléments de Inc (au nombre majoré par $(vl+N)(v+1)^{p+1}$). D'après le paragraphe précédent sur la complexité du calcul des transversaux minimaux, on devrait avoir un nombre de transversaux minimaux borné par $v^{(vl+N)(v+1)^{p+1}}$. Or cette borne supérieure est bien supérieure à 2^v qui est le nombre de parties de $\mathcal{S}_\mathcal{T}$ et donc une borne supérieure du nombre de transversaux minimaux possibles. On voit donc qu'en accumulant les imprécisions dans l'évaluation des bornes supérieures successives, on obtient d'autres bornes qui ne sont plus significatives du tout. Pour continuer l'étude de complexité, on décide donc de poser $\mathcal{I}_e$ comme le nombre d'inconsistances explicites, c'est-à-dire le nombre d'ensembles de S_i de $\mathcal{S}_{\mathcal{T}_\mathcal{Q}}$ minimaux par rapport à l'inclusion tels que leur conjonction est inconsistante, pour pouvoir majorer la cardinalité et le temps de calcul de S_{cons} en fonction de cette valeur. D'après les résultats précédents, on a $\mathcal{I}_e \leq (vl+N)(v+1)^{p+1}$, ou encore $\mathcal{I}_e \leq lN(v+1)^{p+2}$. A partir de $\mathcal{I}_e$, on peut déduire les bornes supérieures de S_{cons} : $\mathcal{H}_{Inc}$ a comme sommets les S_i de $\mathcal{S}_\mathcal{T}$ (au nombre de v) et comme arêtes les $\mathcal{I}_e$ ensembles de S_i de $\mathcal{S}_\mathcal{T}$ contenant une inconsistance explicite. Ainsi, la cardinalité maximale de S_{cons} est $v^{\mathcal{I}_e}$ et le temps de calcul est borné par $(\mathcal{I}_e+1)v^{\mathcal{I}_e}+v.v^{\mathcal{I}_e}$ (le deuxième terme de la somme vient du fait qu'on prend les complémentaires dans $\mathcal{S}_\mathcal{T}$ des transversaux minimaux de $\mathcal{H}_{Inc}$), valeur que l'on peut majorer par $(\mathcal{I}_e+2)v^{\mathcal{I}_e+1}$ ou encore par $\mathcal{I}_e v^{\mathcal{I}_e+1}$. En résumé on a :

- $\mathcal{I}_e \leq lN(v+1)^{p+2}$,
- $|S_{cons}| \leq v^{\mathcal{I}_e}$ et
- le calcul de S_{cons} est $\mathcal{O}(\mathcal{I}_e v^{\mathcal{I}_e+1})$.

Pour terminer l'étude de complexité de la partie de *compute*$\mathcal{ALN}$*BCov* consacrée à la découverte des inconsistances, il reste à évaluer la cardinalité et le temps de calcul de C_{dir} et C_{indir}. Pour ce faire, on dénombre tous les couples (c, H) où H est un ensemble de S_i minimal par rapport à l'inclusion tel que c est une clause de la conjonction normalisée de ces S_i. On sait que ces couples peuvent être de deux types : soit H est constitué d'un seul S_i et c est donc

une clause de ce S_i normalisé, soit H est constitué de plusieurs S_i et alors c est une clause d'exclusion issue d'une inconsistance implicite présente dans les S_i de H. Les couples où H est constitué de plusieurs S_i sont les mêmes couples que les couples (X, c_X) (où $X = H$ et $c_X = c$) dans *compute*$\mathcal{ALN}$*BCov*, regroupés dans l'ensemble $\mathcal{X}$. Il y a en tout v S_i et le nombre maximal de clauses dans un S_i est l, donc il y a au plus $v * l$ couples (c, H) où H est constitué d'un seul S_i. Concernant les couples (X, c_X), on a vu précédemment que le nombre d'inconsistances implicites calculées est majoré par $(vl + N)p(v + 1)^p$. On garde donc cette même borne supérieure pour le nombre de couples (X, c_X). Parmi tous les couples (X, c_X), c'est-à-dire parmi toutes les inconsistances implicites trouvées, certains X contiennent Q. Comme on peut le voir dans l'algorithme, on supprime ces couples avant de former C_{dir} et C_{indir} car les inconsistances implicites correspondantes ne servent à rien. Un raisonnement semblable à celui grâce auquel on a trouvé $(vl + N)p(v + 1)^p$ comme borne supérieure du nombre d'inconsistances implicites nous amène à trouver $vlpv^p \leq lv^{p+2}$ (en majorant p par v) comme borne supérieure du nombre d'inconsistances implicites dans des ensembles de S_i ne contenant pas Q. On prendra donc cette valeur comme nouvelle borne supérieure du nombre de (X, c_X) considérés par la suite. A l'instar de $\mathcal{I}_e$, on nomme $\mathcal{I}_i$ le nombre des inconsistances implicites, c'est-à-dire ce nombre de couples (X, c_X) tels que $X \subseteq \mathcal{S}_\mathcal{T}$ (i.e. $Q \notin X$) et c_X est une clause d'exclusion issue d'une inconsistance implicite dans X. Ainsi, au lieu d'utiliser la borne supérieure lv^{p+2} du nombre d'inconsistances implicites, on utilisera $\mathcal{I}_i$ pour faciliter certaines études de complexité à venir, notamment celles de C_{dir} et C_{indir}. En résumé, on a :

- le nombre de couples (c, H), avec H constitué d'un seul S_i, est $\leq vl$ et
- le nombre de couples (X, c_X) est $\mathcal{I}_i \leq lv^{p+2}$.

Comme, dans chaque couple (c, H) (avec H constitué d'un ou plusieurs S_i), c peut potentiellement couvrir toute clause c_Q de Q, et comme on a N clauses dans Q, on a $|C_{dir}| + |C_{indir}| \leq N(vl + \mathcal{I}_i)$, soit, en remplaçant $\mathcal{I}_i$ par sa borne supérieure et en majorant p par v, $|C_{dir}| \leq N(vl + \mathcal{I}_i) \leq 2Nlv^{p+2}$ et $|C_{indir}| \leq N(vl + \mathcal{I}_i) \leq 2Nlv^{p+2}$.

Intéressons-nous maintenant aux temps nécessaires pour les calculs de C_{dir} et C_{indir}. Il faut dans un premier temps supprimer de l'ensemble $\mathcal{X}$ les couples (X, c_X) tels que $Q \in X$: pour ce faire, il faut examiner cet ensemble $\mathcal{X}$ contenant au plus $(vl + N)p(v + 1)^p$ couples (X, c_X) où X contient au plus $v+1$ S_i, soit un temps de $\mathcal{O}((v+1)(vl+N)p(v+1)^p)$. Après avoir enlevé Q, il reste au plus $\mathcal{I}_i$ couples (X, c_X) comme on l'a vu précédemment. Il faut ensuite ne garder que les (X, c_X) qui possèdent les c_X minimales par rapport à la subsomption (pour un même X) : on peut faire cette sélection en comparant 2 à 2 les couples, ce qui se fait en un temps $\mathcal{O}(p{\mathcal{I}_i}^2)$ (le facteur p vient du fait que le test de subsomption entre deux clauses dépend de la profondeur de celles-ci). Enfin, il faut comparer toutes les clauses c des couples (c, H) où H est constitué d'un ou plusieurs S_i avec toutes les clauses c_Q de Q (au plus N) pour voir si elles couvrent les c_Q et comment (directement ou indirectement) : on a au plus vl couples (c, H) où H est constitué d'un seul S_i, au plus $\mathcal{I}_i$ couples (c, H) où H est constitué de plusieurs S_i, donc, en comptant p un test de couverture directe ou indirecte, on obtient un temps de $\mathcal{O}(pN(vl + \mathcal{I}_i))$. En tout, le temps de calcul de C_{dir} et C_{indir} est de $\mathcal{O}((v+1)(vl+N)p(v+1)^p + p{\mathcal{I}_i}^2 + pN(vl+\mathcal{I}_i))$, d'où en remplaçant $\mathcal{I}_i$ par sa borne supérieure : $\mathcal{O}((v+1)(vl+N)p(v+1)^p + pl^2v^{2(p+2)} + pN(vl+lv^{p+2}))$ soit $\mathcal{O}(4Nl^2(v + 1)^{2p+4})$. En résumé on a :

- $\mathcal{I}_i \leq lv^{p+2}$,
- $|C_{dir}| \leq N(vl + \mathcal{I}_i) \leq 2Nlv^{p+2}$,
- $|C_{indir}| \leq N(vl + \mathcal{I}_i) \leq 2Nlv^{p+2}$ et
- le temps de calcul de C_{dir} et C_{indir} est $\mathcal{O}(l^2N(v + 1)^{2p+4})$.

Complexité du calcul des couvertures de rest maximal

Pour calculer les couvertures de rest maximal, il faut calculer S_{couv} et S_{rest}, ce qui permet par la suite d'obtenir les ensembles des plus grands et des plus petits (par rapport à l'inclusion) ensembles de S_i correspondant à des couvertures de rest maximal. Pour avoir toutes les autres couvertures de rest maximal, il faut ensuite énumérer tous les ensembles de S_i contenus dans une plus grande et contenant une plus petite couverture de rest maximal.

S_{couv} est un sous-ensemble de S_{cons}, donc au pire sa cardinalité est celle de S_{cons}, soit $v^{\mathcal{I}_e}$. On calcule S_{couv} en testant l'inclusion entre tout H d'un triplet (c, H, c_Q) de $C_{dir} \cup C_{indir}$ et tout $E \in S_{cons}$. On a en tout $vl + \mathcal{I}_i$ couples (c, H), et N clauses c_Q. Sachant que H et E sont des ensembles d'au plus v S_i supposés triés, le temps de calcul de S_{couv} est donc $\mathcal{O}(N(vl + \mathcal{I}_i)v.v^{\mathcal{I}_e})$, soit $\mathcal{O}(Nl\mathcal{I}_i v^{\mathcal{I}_e+2})$, soit, en remplaçant $\mathcal{I}_i$ par sa borne supérieure lv^{p+2}, un temps en $\mathcal{O}(l^2 N v^{\mathcal{I}_e+p+4})$. En résumé, on a :

- $|S_{couv}| \leq v^{\mathcal{I}_e}$ et
- le calcul de S_{couv} est $\mathcal{O}(l^2 N v^{\mathcal{I}_e+p+4})$.

A ce stade de l'algorithme, on connaît les bornes supérieures et inférieures de l'espace de recherche des couvertures, c'est-à-dire que l'on connaît les plus grands (resp. les plus petits) ensembles de S_i qui contiennent (resp. sont contenus dans) les couvertures. Les bornes supérieures des cardinalités des ensembles précédents étant larges, il est difficile d'en déduire une borne supérieure précise du nombre de couvertures de rest maximal. On se contentera donc de rappeler que ce nombre est majoré par 2^v. Pour obtenir la liste de toutes les couvertures, il faut énumérer, pour chaque plus grand ensemble de S_i de la borne supérieure (au nombre de $|S_{couv}| \leq v^{\mathcal{I}_e}$), toutes les sous-parties (au plus 2^v) en ne gardant que celles qui contiennent (test d'inclusion en $\mathcal{O}(v)$) au moins un plus petit ensemble de $C_{dir} \cup C_{indir}$ (au nombre majoré par $2Nlv^{p+2}$). Cette énumération et ces tests se font globalement en un temps de $\mathcal{O}(v^{\mathcal{I}_e} * 2^v * 2Nlv^{p+2} * v)$. En résumé, on a :

- le nombre de couvertures est $|cov(\mathcal{T}, Q)| \leq 2^v$
- le temps d'énumération de toutes les couvertures est en $\mathcal{O}(lNv^{\mathcal{I}_e+p+3} * 2^v)$.

En tant que sous-ensemble de S_{couv}, la cardinalité de S_{rest} est majorée par celle de S_{couv}, soit $v^{\mathcal{I}_e}$. Le calcul de S_{rest} se fait en comparant, pour chaque couple (E'_1, E'_2) d'éléments de S_{couv}, les rest associés, et ceci en comparant les clauses de $\widehat{\mathcal{G}}^{\#}_{E'_1}$ et de $\widehat{\mathcal{G}}^{\#}_{E'_2}$ d'après le lemme 4.3.3 page 65.

La première étape est donc de d'obtenir $\widehat{\mathcal{G}}^{\#}_{E'_i}$ pour E'_i élément de S_{couv}, c'est-à-dire qu'il faut normaliser tous les éléments de S_{couv}. Comme toute clause d'une conjonction normalisée de S_i provient d'un S_i ou d'une inconsistance implicite et que l'on a calculé toutes ces inconsistances implicites, alors on a dans C_{dir} et C_{indir} toutes les clauses possibles que l'on peut trouver dans $\widehat{\mathcal{G}}^{\#}_{E'_i}$. Ces clauses sont au nombre de vl pour les clauses provenant directement des S_i plus vlp pour les clauses d'exclusion c_X présentes dans les $\mathcal{I}_i$ couples (X, c_X). Cette valeur vlp provient des calculs de transversaux minimaux initiaux : pour toute clause c des S_i (au plus vl), on construit un hypergraphe dont les arêtes correspondent aux p (au plus) clauses de l'approximation faible de sa négation ; il en résulte un nombre total de clauses d'exclusion possibles d'au plus vlp[40]. Pour

[40] Cette valeur vlp est à mettre en rapport avec $\mathcal{I}_i \leq lv^{p+1}$ qui borne le nombre de couples (X, c_X). $\mathcal{I}_i$ est plus grand que vlp car une même clause d'exclusion peut être présente dans plusieurs X différents.

obtenir l'ensemble des clauses normalisées d'un E'_i, on commence par regrouper toutes les clauses des S_i de E'_i (au plus vl clauses), puis on ajoute les clauses c_X de tous les couples (X, c_X) de $\mathcal{X}$ tels que $X \subseteq E'_i$ (au plus $\mathcal{I}_i$ couples et v pour un test d'inclusion entre X et E'_i, soit un temps borné par $v\mathcal{I}_i$). Ensuite, pour ne garder que les plus petites clauses par rapport à la subsomption, on compare par rapport à la subsomption tous les couples de clauses de l'ensemble obtenu (au plus $vl + vlp$ clauses en comptant p pour un test de subsomption entre deux clauses, soit un temps borné par $p(vl+vlp)^2$). Au total on obtient un temps de calcul $\mathcal{O}(vl+v\mathcal{I}_i+p(vl+vlp)^2) = \mathcal{O}(v^2(l+1)^2(p+1)^3(\mathcal{I}_i+1))$ pour un E'_i de S_{couv}. La normalisation de tous les éléments de S_{couv} est donc $\mathcal{O}(v^2(l+1)^2(p+1)^3(\mathcal{I}_i+1)v^{\mathcal{I}_e}) = \mathcal{O}((l+1)^2(p+1)^3(\mathcal{I}_i+1)v^{\mathcal{I}_e+2})$.

La deuxième étape est l'application du lemme 4.3.3 page 65. Pour appliquer ce lemme, on commence par déterminer pour chaque E'_i de S_{couv} (au plus $v^{\mathcal{I}_e}$) l'ensemble A_i des clauses c_Q de Q couvertes directement par E'_i, et l'ensemble B_i des clauses c_Q de Q couvertes indirectement par E'_i. On trouve A_i et B_i en comparant chaque clause de E'_i (au plus $vl + vlp$) avec chaque clause de Q (au plus N, avec un temps de comparaison d'au plus p). Le calcul de A_i et B_i pour tous les E'_i de S_{couv} est donc $\mathcal{O}(pN(vl + vlp)v^{\mathcal{I}_e})$. Ensuite, pour chaque couple (E'_1, E'_2) d'éléments de S_{couv} (au plus $v^{2\mathcal{I}_e}$ couples), on teste si $Rest_{E'_2}(Q) \sqsubseteq Rest_{E'_1}(Q)$. Pour ce faire, on teste d'abord si $A_2 \cup B_2 \subseteq A_1 \cup B_1$, en un temps $\mathcal{O}(2Np)$, car p est le temps maximal pour tester l'égalité entre deux clauses et $|A_i \cup B_i| \leq N$ puisque A_i et B_i sont des ensembles de clauses de Q et que l'on a N clauses dans Q. On regarde ensuite pour chaque clause c_Q de B_1 (au plus N) toutes les clauses $c_{E'_2}$ de E'_2 (au plus $vl + vlp$) en testant si c_Q est couverte directement, indirectement ou non couverte par $c_{E'_2}$ (test au plus p), et si c_Q est couverte indirectement alors on examine toutes les clauses $c_{E'_1}$ de E'_1 (au plus $vl+vlp$) pour voir si $c_{E'_1} \sqsubseteq c_{E'_2}$ (test au plus p). Le temps de calcul pour un couple (E'_1, E'_2) est donc $\mathcal{O}(2Np + Np^2(vl + vlp)^2)$. D'où un temps $\mathcal{O}((2Np + Np^2(vl + vlp)^2)v^{2\mathcal{I}_e})$, soit $\mathcal{O}((2Np^2(vl + vlp)^2)v^{2\mathcal{I}_e}) = \mathcal{O}(2Nl^2(p+1)^4v^{2(\mathcal{I}_e+1)})$ pour tous les couples. Au final, le temps mis pour comparer les rest est $\mathcal{O}(3Nl^2(p+1)^4v^{2(\mathcal{I}_e+1)})$.

On déduit des deux paragraphes précédents que le temps de calcul de S_{rest} est $\mathcal{O}((l+1)^2(p+1)^3(\mathcal{I}_i+1)v^{\mathcal{I}_e+2})$ pour la normalisation et $\mathcal{O}(3Nl^2(p+1)^4v^{2(\mathcal{I}_e+1)})$ pour la comparaison des rest. Le temps total d'obtention de S_{rest} est de $\mathcal{O}(Nl^2p^4\mathcal{I}_iv^{2\mathcal{I}_e+2})$. En remplaçant $\mathcal{I}_i$ par sa borne supérieure lv^{p+2}, on obtient $\mathcal{O}(Nl^3p^4v^{2\mathcal{I}_e+p+4})$. En résumé, on a :

- $|S_{rest}| \leq v^{\mathcal{I}_e}$ et
- le calcul de S_{rest} est $\mathcal{O}(Nl^3p^4v^{2\mathcal{I}_e+p+4})$.

Nous étudions maintenant les bornes supérieures des cardinalités des ensembles $C_{dir}(E^*)$, $C_{indir}(E^*)$ et $C_{egal}(E^*)$, pour un E^* élément de S_{rest} donné. $C_{dir}(E^*)$ est un sous-ensemble de C_{dir}. Donc $|C_{dir}(E^*)| \leq |C_{dir}| \leq N(vl + \mathcal{I}_i)$. De même pour $C_{indir}(E^*)$, on a $|C_{indir}(E^*)| \leq |C_{indir}| \leq N(vl + \mathcal{I}_i)$. $C_{egal}(E^*)$ est l'union d'un ensemble de couples (Y, c_Q) extrait des triplets de $C_{dir}(E^*)$, donc de cardinalité $\leq N(vl + \mathcal{I}_i)$, et d'un ensemble de couples (Y, c_Q) où c_Q est dans au moins un triplet de $C_{indir}(E^*)$ et où Y est calculé par un produit cartésien. En tout, un produit cartésien est calculé pour chaque clause c_Q de Q (au plus N). Les termes du produit cartésien sont au nombre des clauses $\widetilde{c}$ de E^* (au plus $vl + vlp = vl(p+1)$) et sont les ensembles de $\widetilde{H}$ tels que $(\widetilde{c}, \widetilde{H}, c_Q)$ est dans $C_{indir}(E^*)$. Comme on a au plus $vl(p + 1)$ clauses $\widetilde{c}$ et au plus $vl + \mathcal{I}_i$ ensembles $\widetilde{H}$ (et pas $N(vl + \mathcal{I}_i)$ car c_Q est fixée), on a au plus, pour une clause c_Q, $(vl + \mathcal{I}_i)^{vl(p+1)}$ ensembles Y possibles, soit $N(vl + \mathcal{I}_i)^{vl(p+1)}$ pour toutes les clauses c_Q de Q. Cependant, cette borne est bien au-delà du nombre de couples (Y, c_Q) que l'on peut avoir dans $C_{egal}(E^*)$. En effet, après le produit cartésien, les tuples sont fusionnés pour obtenir des ensembles de S_i, et non des ensembles d'ensembles de S_i. Ainsi, au maximum, on ne peut avoir

dans $C_{egal}(E^*)$ que 2^v ensembles Y pour chaque c_Q de Q, c'est-à-dire au plus $N2^v$ couples (Y, c_Q). Cette valeur $N2^v$ englobe les $N(vl + \mathcal{I}_i)$ couples issus du premier ensemble définissant $C_{egal}(E^*)$. En résumé, on a :

- $|C_{dir}(E^*)| \leq N(vl + \mathcal{I}_i)$,
- $|C_{indir}(E^*)| \leq N(vl + \mathcal{I}_i)$,
- $|C_{egal}(E^*)| \leq N2^v$
- et ce pour $\leq v^{\mathcal{I}_e}$ ensembles E^*.

Intéressons-nous maintenant aux temps de calcul de ces ensembles. Pour calculer $C_{dir}(E^*)$, il faut tester l'inclusion (en temps au plus v) entre tous les H des triplets de C_{dir} et E^*. S'il y a au plus $N(vl + \mathcal{I}_i)$ triplets dans C_{dir}, il n'y a au plus que $(vl + \mathcal{I}_i)$ H différents dans ces triplets. On a donc un temps de calcul de $C_{dir}(E^*)$ qui est en $\mathcal{O}(v(vl + \mathcal{I}_i))$, soit $\mathcal{O}(l\mathcal{I}_i v^2)$. En remplaçant $\mathcal{I}_i$ par sa borne supérieure lv^{p+2}, on obtient $\mathcal{O}(l^2 v^{p+4})$.

Pour calculer $C_{indir}(E^*)$, il faut calculer les mêmes inclusions que pour $C_{dir}(E^*)$ et en plus trier les clauses c (présentes dans les triplets de C_{indir} sélectionnés par le test d'inclusion) par rapport à la subsomption pour ne garder que les plus petites : on a au plus $vl + vlp = vl(p + 1)$ clauses c, on les compare 2 à 2 en au plus $v^2 l^2 (p + 1)^2$ et on teste à chaque fois la subsomption en au plus p, soit un temps $\mathcal{O}(v^2 l^2 (p + 1)^3)$. Au final, le calcul de $C_{indir}(E^*)$ est $\mathcal{O}(v(vl + \mathcal{I}_i) + v^2 l^2 (p + 1)^3)$, soit $\mathcal{O}(3\mathcal{I}_i v^2 l^2 (p + 1)^3)$, ou encore $\mathcal{O}(3 * \mathcal{I}_i v^2 l^2 (2p)^3)$. En remplaçant $\mathcal{I}_i$ par sa borne supérieure lv^{p+2}, on obtient $\mathcal{O}(24 l^3 p^3 v^{p+4})$.

Le premier ensemble de l'union définissant $C_{egal}(E^*)$ est calculé en au plus $N(vl+\mathcal{I}_i)$ puisqu'il résulte de l'extraction des couples (H, c_Q) des triplets de $C_{dir}(E^*)$ qui sont au nombre de $N(vl+\mathcal{I}_i)$ (au plus). Le deuxième ensemble est calculé en quatre étapes répétées pour toutes les clauses c_Q de Q (au plus N) :

- Il faut construire les ensembles dont on fera ensuite le produit cartésien : pour chaque clause $\widetilde{c}$ de E^* (au plus $vl(p + 1)$), pour chaque triplet (c, H, c_Q) de $C_{indir}(E^*)$ (au plus $(vl + \mathcal{I}_i)$, et pas $N(vl + \mathcal{I}_i)$ puisque c_Q est fixée), on teste l'égalité des clauses $\widetilde{c}$ et c (au plus p) ; au total, on a un temps d'au plus $pvl(p + 1)(vl + \mathcal{I}_i)$.
- Il faut calculer le produit cartésien, soit un temps d'au plus $(vl + \mathcal{I}_i)^{vl(p+1)}$ pour obtenir autant de tuples.
- Il faut fusionner les tuples obtenus : dans chaque tuple (au plus $(vl + \mathcal{I}_i)^{vl(p+1)}$), on a au plus $vl(p + 1)$ ensembles de S_i (dont la cardinalité est au plus de v) à fusionner. En supposant que les ensembles de S_i soient chacun triés, la fusion est linéaire par rapport au nombre totale de S_i, soit $v^2 l(p + 1)$. D'où un temps d'au plus $v^2 l(p + 1)(vl + \mathcal{I}_i)^{vl(p+1)}$ pour tous les tuples. En majorant $(p + 1)$ par v on obtient $(vl + \mathcal{I}_i)^{vl(p+1)+3}$.
- Il faut enfin minimiser les ensembles de S_i par rapport à l'inclusion. On a vu précédemment que ceux-ci étaient au plus au nombre de 2^v, donc on les teste deux à deux (au plus 2^{2v} tests) par rapport à l'inclusion (au plus v pour un test), soit un temps d'au plus $v2^{2v}$.

On répète ces quatre étapes pour les N clauses c_Q de Q, d'où un temps d'au plus $N[pvl(p + 1)(vl+\mathcal{I}_i)+(vl+\mathcal{I}_i)^{vl(p+1)}+(vl+\mathcal{I}_i)^{vl(p+1)+3}+v2^{2v}]$, soit un temps en $\mathcal{O}(4(vl+\mathcal{I}_i)^{vl(p+1)+3})$ (car le terme en v^v est bien plus grand que tous les autres), soit $\mathcal{O}(4(vl + \mathcal{I}_i)^{2vlp+3})$. En remplaçant $\mathcal{I}_i$ par sa borne supérieure lv^{p+2}, on obtient $\mathcal{O}((2lv^{p+2})^{2vlp+3})$. C'est une borne supérieure du temps total de calcul de $C_{egal}(E^*)$. En résumé, on a :

- le temps de calcul de $C_{dir}(E^*)$ est $\mathcal{O}(l^2 v^{p+4})$,
- le temps de calcul de $C_{indir}(E^*)$ est $\mathcal{O}(l^3 p^3 v^{p+4})$,
- le temps de calcul de $C_{egal}(E^*)$ est $\mathcal{O}((2lv^{p+2})^{2vlp+3})$
- et ce pour $\leq v^{\mathcal{I}_e}$ ensembles E^*.

Une fois l'ensemble $C_{egal}(E^*)$ obtenu, il s'agit de calculer les transversaux minimaux de l'hypergraphe $\mathcal{H}_{E^*}$ construit à partir de $C_{egal}(E^*)$: les sommets sont les Y des couples (Y, c_Q) de $C_{egal}(E^*)$, au nombre maximum de 2^v, et les arêtes sont les c_Q des couples (Y, c_Q) de $C_{egal}(E^*)$, soit au plus N. Le nombre de transversaux minimaux est donc d'au plus 2^{vN}. Or ces transversaux minimaux sont des ensembles d'ensembles de S_i qui sont fusionnés (puis minimisés par rapport à l'inclusion) en des ensembles de S_i. Donc, au total, on aura au plus 2^v éléments dans $R_{eq}(E^*)$. En ce qui concerne le temps de calcul de ces éléments, il est divisé en trois : le temps de calcul des transversaux minimaux, celui de la fusion et celui de la minimisation. Le temps de calcul des transversaux minimaux est en $\mathcal{O}((N+1)2^{vN})$. La fusion d'un transversal minimal revient à un parcours de chacun de ses ensembles de S_i. Au plus un transversal minimal est constitué de N ensembles de S_i (car $\mathcal{H}_{E^*}$ possède N arêtes, et chacun de ces ensembles de S_i a une cardinalité d'au plus v. La fusion d'un transversal minimal est donc $\mathcal{O}(vN)$, et celle de tous les transversaux minimaux est $\mathcal{O}(vN2^{vN})$. Enfin la minimisation consiste en un test d'inclusion en $\mathcal{O}(v)$ pour chaque couple de transversax minimaux fusionnés, soit au plus 2^{2v} couples. Ainsi, la minimisation est en $\mathcal{O}(v2^{2v})$. Au final, l'obtention de $R_{eq}(E^*)$ est en $\mathcal{O}((N+1)2^{vN} + vN2^{vN} + v2^{2v}) = \mathcal{O}(vn2^{vN})$. En résumé, on a :

- $|R_{eq}(E^*)| \leq 2^v$,
- le temps de calcul de $R_{eq}(E^*)$ est $\mathcal{O}(vN2^{vN})$
- et ce pour $\leq v^{\mathcal{I}_e}$ ensembles E^*.

A ce stade de l'algorithme, on connaît les bornes supérieures et inférieures de l'espace de recherche des couvertures de rest maximal, c'est-à-dire que l'on connaît les plus grands (resp. les plus petits) ensembles de S_i qui contiennent (resp. sont contenus dans) les couvertures de rest maximal. Les bornes supérieures des cardinalités des ensembles précédents étant larges, il est difficile d'en déduire une borne supérieure précise du nombre de couvertures de rest maximal. On se contentera donc de rappeler que ce nombre est majoré par 2^v. Pour obtenir la liste de toutes les couvertures de rest maximal, il faut énumérer, pour chaque plus grand ensemble de S_i de la borne supérieure (au nombre de $|S_{rest}| \leq v^{\mathcal{I}_e}$), toutes les sous-parties (au plus 2^v) en ne gardant que celles qui contiennent (test d'inclusion en $\mathcal{O}(v)$) au moins un plus petit ensemble de $R_{eq}(E^*)$ (au nombre majoré par 2^v). Cette énumération et ces tests se font globalement en un temps de $\mathcal{O}(v^{\mathcal{I}_e} * 2^v * 2^v * v)$. En résumé, on a :

- le nombre de couvertures de rest maximal est $|cov_{rest}(\mathcal{T}, Q)| \leq 2^v$
- le temps d'énumération de toutes les couvertures de rest maximal est en $\mathcal{O}(v^{\mathcal{I}_e+1} * 2^{2v})$.

Complexité du calcul des meilleures couvertures

Pour calculer les meilleures couvertures, il faut calculer les miss des couvertures de rest maximal, les comparer deux à deux, ne garder que les couvertures de miss maximal puis comparer toutes les couvertures restantes (qui ont donc un rest et un miss maximal) par rapport à l'inclusion.

Comme on l'a vu dans l'exemple 19 page 68, les miss de toutes les couvertures de rest maximal doivent être calculés car une couverture incluse dans une autre n'a pas forcément un miss plus petit ou plus grand. Donc il faut parcourir toutes les couvertures de rest maximal, comme on l'a vu précédemment, et calculer à chaque fois le miss correspondant. Le calcul des miss de toutes les couvertures de rest maximal est décomposé en trois étapes :

– Normalisation des ensembes de S_i qui sont des couvertures de rest maximal.

Il y a au plus 2^v couvertures de rest maximal, et, comme on a vu précédemment, la normalisation d'un ensemble de S_i est en $\mathcal{O}(v^2(l+1)^2(p+1)^3(\mathcal{I}_i+1))$. Au total, normaliser toutes les couvertures de rest maximal est $\mathcal{O}(l^3p^3v^{p+4}2^v)$.

– Calcul des lcs.
Le calcul d'un lcs se fait en comparant toutes les clauses de la couverture de rest maximal courante (au plus $vl(p+1)$ clauses) avec toutes celles de Q (au plus N) par rapport à la subsomption (en un temps $\mathcal{O}(v)$). Ainsi, le calcul de tous les lcs est borné par $\mathcal{O}(lNpv^2 2^v)$.

– Calcul des différences sémantiques.
Pour le calcul d'une différence sémantique $E - Q$, on a les étapes suivantes :
  – pour toutes les clauses de E (au plus $vl(p+1)$), pour toutes les clauses de Q (au plus N), on teste si on est dans les cas d'une inconsistance implicite (au pire on est toujours dans ce cas) : si oui, on calcule l'approximation faible de la négation d'une clause (en au plus p), on teste par rapport à la subsomption (au plus p) chaque clause de l'approximation (au plus p) avec chaque clause de Q (au plus N). Cette étape se fait donc en un temps $\mathcal{O}(vl(p+1) * pN * p * p^2N)$, soit $\mathcal{O}(lNp^5v)$.
  – Il faut alors générer les descriptions résultats de la différence en combinant les clauses issues de l'étape précédente. Chaque clause de E génère (dans le cas des inconsistances implicites) plusieurs clauses. La somme de ces clauses générées ne peut pas dépasser le nombre N de clauses de Q puisque l'on a au pire une clause générée par clause de Q. Pour avoir une borne supérieure du nombre de descriptions que ces clauses générées peuvent foremr par combinaisons, on peut majorer, pour chaque clause de E, le nombre de clauses générées par N. Ainsi la borne supérieure du nombre de descriptions résultats de la différence sémantique est $N^{vl(p+1)}$.

  Au total, le temps de calcul d'une différence sémantique est en $\mathcal{O}(lNp^5v+N^{vl(p+1)})$, soit, en majorant $Nlpv$ par $N^{vl(p+1)}$, $\mathcal{O}(p^4N^{vl(p+1)})$. Pour toutes les couvertures de rest maximal, on obtient donc $\mathcal{O}(p^4N^{vl(p+1)}2^v)$.

Ainsi le calcul des miss se fait en un temps $\mathcal{O}(l^3p^3v^{p+4}2^v + lNpv^2 2^v + p^4N^{vl(p+1)}2^v)$. En majorant $lp2^v$ par $N^{2vl(p+1)}$ dans le premier terme, $lNp2^v$ par $N^{2vl(p+1)}$ dans le second et $p2^v$ par $N^{vl(p+1)}$ dans le troisième, on obtient $\mathcal{O}((l^2p^2v^{p+4}+v^2+p^3)N^{2vl(p+1)})$, soit $\mathcal{O}(l^2p^2v^{p+4}N^{2vl(p+1)})$ pour le calcul de tous les miss.

Après le calcul de tous les miss, on les compare deux à deux par rapport à la subsomption entre ensembles de descriptions pour ne garder que les plus grands.Pour tout couple de couvertures de rest maximal (au plus 2^{2v} couples), pour tout couples de descriptions de concepts des miss correspondants (au plus $N^{vl(p+1)}$ descriptions par miss, donc au plus $N^{2vl(p+1)}$ couples de descriptions), pour tout couples de clauses des descriptions courantes (au plus $(vl(p+1))^2$), on fait un test de subsomption (au plus p). Au final, la comparaison des miss deux à deux est en $\mathcal{O}(2^{2v} * N^{2vl(p+1)} * (vl(p+1))^2 * p)$. En majorant 2^{2v} par $N^{2vl(p+1)}$ on obtient $\mathcal{O}(l^2p^3v^2N^{2vl(p+1)})$.

La dernière étape pour l'obtention des meilleures couvertures est le tri des couvertures de rest et miss maximal par rapport à l'inclusion, puisque les meilleures couvertures sont celles qui sont les plus petites par rapport à l'inclusion. On a au plus 2^v couvertures de rest et de miss maximal. On va les comparer deux à deux, en sachant qu'une comparaison se fait en un temps d'au plus v (puisqu'il y a au plus v S_i dans une couverture). On obtient alors $\mathcal{O}(v2^{2v})$ pour cette étape de minimisation par rapport à l'inclusion.

En résumé, on a :
- le calcul de tous les miss se fait en un temps $\mathcal{O}(l^2p^2v^{p+4}N^{2vl(p+1)})$,

- la comparaison des miss deux à deux se fait en un temps $\mathcal{O}(l^2p^3v^2N^{2vl(p+1)})$ et
- la minimisation des couvertures par rapport à l'inclusion se fait en un temps de $\mathcal{O}(v2^{2v})$.

Annexe C

Justification de l'intérêt du théorème des persistants

A la section 3.2, nous avons étudié l'algorithme *computeBCov*. Nous avons vu qu'il était basé sur le calcul des transversaux minimaux dans un hypergraphe. Pour ce calcul nous avons rappelé l'algorithme classique, l'algorithme 1 page 34, et nous avons donné l'algorithme 2 page 37, basé sur le théorème 3.2.1 des persistants (page 35), dont nous avons dit qu'il en était une amélioration. Nous en donnons dans cette annexe la justification précise.

Pour justifier que l'algorithme 2 améliore l'algorithme 1, nous proposons d'étudier l'évolution des nombres d'opérations élémentaires (tests d'inclusion et calculs d'intersection entre ensembles de sommets, et nombre de transversaux minimaux générés à chaque itération) pour des cas très mauvais qui maximisent ces quantités pour les deux algorithmes. C'est-à-dire que l'on cherche à construire deux hypergraphes H_1 et H_2 qui sont de très mauvais cas pour, respectivement, l'algorithme 1 et l'algorithme 2. Ainsi, H_1 doit maximiser les nombres d'opérations élémentaires à chaque itération de l'algorithme 1, et de même pour H_2 avec l'algorithme 2. On peut ainsi comparer le comportement au pire des deux algorithmes.

Après quelques remarques sur la notion d'opérations élémentaires dans la section C.1, nous étudions, dans la section C.2, les caractéristiques d'un très mauvais cas pour l'algorithme 1. Nous en déduisons l'hypergraphe H_1. Dans la section C.3, nous étudions les caractéristiques d'un très mauvais cas pour l'algorithme 2, pour en déduire l'hypergraphe H_2. Enfin, dans la section C.4, nous comparons l'exécution avec H_1 et H_2 des deux algorithmes.

C.1 Opérations élémentaires

Supposons que nous avons un hypergraphe $\mathcal{H}$ avec m arêtes et n sommets. Nous examinons deux sortes d'opérations élémentaires :

- Les tests d'inclusion et calcul d'intersection entre deux ensembles de sommets : comme le nombre maximum de sommets qu'il peut y avoir dans un ensemble de sommets est n, alors tester une inclusion entre deux ensembles de sommets a une complexité de $\mathcal{O}(n^2)$. Si les ensembles sont ordonnés, alors la complexité est $\mathcal{O}(n)$. Comme le calcul d'une intersection entre deux ensembles de sommets a la même complexité qu'un test d'inclusion, on regroupe tests d'inclusion et calcul d'intersections dans un même type d'opérations élémentaires.
- La génération d'un transversal comme union d'un transversal (ensemble de sommets) généré à l'itération précédente $i-1$ et d'un sommet de l'arête e examiné à l'itération i. Dans le cas d'un ensemble non ordonné, cette opération est $\mathcal{O}(n)$ si l'on ne sait pas si le

sommet à ajouter appartient ou non au transversal de l'itération $i-1$, et $\mathcal{O}(1)$ (en temps constant) si on sait que le sommet à ajouter n'est pas dans le transversal de l'itération i (on peut alors l'ajouter à la fin puisque l'ensemble n'est pas ordonné). On sait si un sommet appartient à l'ensemble notamment dans l'algorithme 2 puisque l'on ne génère de nouveaux transversaux minimaux qu'à partir des non-persistants qui ont une intersection vide avec l'arête courante. Dans le cas d'un ensemble ordonné, l'ajout d'un sommet est toujours $\mathcal{O}(n)$.

Pour être en accord avec notre implémentation de l'algorithme 2 (voir le chapitre 7) dans laquelle les tests d'inclusions et calculs d'intersection sont $\mathcal{O}(n^2)$ et les générations de transversaux sont $\mathcal{O}(n)$, nous pouvons dire qu'il existe un facteur n entre un test d'inclusion et une génération de transversal. Dans la construction de très mauvais cas pour les algorithmes étudié, on cherchera donc à maximiser avant tout le nombre de tests d'inclusions et de calculs d'intersection à faire.

C.2 Un très mauvais cas pour l'algorithme 1

Dans cette section, on reprend l'algorithme 1 en essayant d'évaluer le nombre maximal d'opérations élémentaires effectuées et le nombre maximal de transversaux générés à chaque itération. A partir de ces valeurs théoriques, on déduit l'hypergraphe H_1 qui les respecte. H_1 est donc un très mauvais cas pour l'algorithme 1.

Rappelons le principe de l'algorithme 1. A chaque itération (une itération par arête de l'hypergraphe), on génère des transversaux minimaux candidats en calculant toutes les unions possibles d'un sommet de l'arête courante avec un transversal minimal de l'itération précédente. Dans un deuxième temps, on supprime tous les transversaux candidats non minimaux. On pose ainsi x_{i-1} comme étant le nombre de transversaux minimaux générés à la fin de l'itération $i-1$, et e_i l'arête examinée à l'iteration i.

A chaque itération, l'algorithme 1 s'exécute en deux étapes :

- L'étape de génération : le nombre de transversaux qui sont générés est toujours $x_{i-1} * |e_i|$.
- L'étape de suppression des candidats non minimaux : pour identifier ceux qui sont minimaux, il est obligatoire de tester si chaque transversal candidat est inclus dans chaque autre. Ainsi, le nombre de tests d'inclusion est au maximum de $(x_{i-1} * |e_i|) * (x_{i-1} * |e_i| - 1)$, avec $|e_i| = \mathcal{O}(n)$.

A la fin de chaque itération on a alors généré au plus $|e_i|$ nouveaux transversaux minimaux pour chacun des x_{i-1} transversaux générés à l'itération précédente.

Le pire cas est donc celui dans lequel chaque arête de l'hypergraphe n'a que des sommets qui sont seulement dans cette arête. C'est le cas classique instancié dans l'exemple ci-dessous et qui implique un nombre exponentiel de transversaux minimaux, ce nombre étant $\mathcal{O}(n^m)$.

Exemple 37

Soit l'hypergraphe H_1 suivant, possédant 13 arêtes et 2 sommets par arête (soit 26 sommets) :

$$H_1 = \{\{1,2\},\{3,4\},\{5,6\},\{7,8\},\{9,10\},\{11,12\},\{13,14\},$$

$$\{15,16\},\{17,18\},\{19,20\},\{21,22\},\{23,24\},\{25,26\}\}$$

L'ensemble des transversaux minimaux $Tr(H_1)$ de H_1 contient $2^{13} = 8192$ transversaux minimaux. Pour l'algorithme 1, le nombre de tests d'inclusion est $\sum_{i=1}^{13} 2^i * (2^i - 1) = 89462102$.

∘

C.3 Un très mauvais cas pour l'algorithme 2

Dans cette section, on adopte la même démarche que dans la section précédente. On évalue le nombre maximal d'opérations élémentaires effectuées et le nombre maximal de transversaux générés à chaque itération. A partir de ces valeurs théoriques, on déduit l'hypergraphe H_2 qui les respecte. H_2 est donc un très mauvais cas pour l'algorithme 2.

Rappelons le principe de l'algorithme 2. Basé sur le théorème des persistants, il s'agit ici de ne générer à chaque itération que les candidats minimaux et aucun candidat non minimal. Pour ce faire, il faut effectuer un certain nombre de tests de vérification, avant la génération de transversaux minimaux. Comme dans l'algorithme 1, la génération consiste en l'union d'un transversal minimal X_{i-1} de l'itération précédente et d'un sommet c_i de l'arête courante e_i. La phase de vérification, préalable à la génération, consiste à vérifier qu'il n'existe aucun autre transversal minimal X'_{i-1} de l'itération précédente qui a pour intersection avec e_i un unique sommet c' tel que $(X'_{i-1} \setminus \{c'\}) \subset X_{i-1}$. La phase de vérification de l'algorithme 2 remplace celle de suppressions des candidats non minimaux de l'algorithme 1.

Ainsi, à l'itération i, il y a trois étapes :

- Etape 1 : les intersections entre les x_{i-1} transversaux minimaux générés à l'itération $i-1$ et l'arête e_i sont calculées afin de déterminer pour chaque transversal minimal généré à l'itération $i-1$ si c'est un 1-, n- ou non-persistant à l'itération i. On a donc x_{i-1} intersections à calculer. Soit, respectivement, r_i, s_i et t_i le nombre de 1-, n-, et non-persistants à l'iteration i.
- Etape 2 : selon le théorème 3.2.1, chaque sommet de e_i dont le singleton est l'intersection de e_i avec un transversal minimal généré à l'itération $i-1$ doit être enlevé de ce transversal afin de tester l'inclusion stricte entre chaque non-persistant et ce transversal. Ainsi, cette étape est une étape de tests d'inclusion.
- Etape 3 : chaque non-persistant génère au maximum $|e_i|$ nouveaux transversaux. C'est donc une étape de générations de transversaux minimaux.

Afin de construire un très mauvais cas, et comme les calculs d'intersections de l'étape 1 sont obligatoires (pour pouvoir implémenter le théorème 3.2.1), une façon simple est de maximiser à la fois le nombre de tests d'inclusion à l'étape 2 et le nombre de générations de transversaux de l'étape 3. Comme nous l'avons vu précédemment, un test d'inclusion est plus coûteux qu'une génération, mais afin d'avoir le plus de test d'inclusion possible, il faut aussi maximiser le nombre de transversaux minimaux générés à chaque itération.

Dans un premier temps, on peut constater que maximiser ces deux nombres (tests d'inclusion et générations de transversaux) revient à assurer qu'il n'y a aucun n-persistant (i. e. $s_i = 0$) à chaque itération. En effet, si $s_i = 0$, $r_i + t_i$ sera maximum et alors le nombre de tests d'inclusion et de générations de transversaux sera plus grand.

En ne regardant que le nombre de tests d'inclusion, nous devons remarquer que le nombre de ces tests à l'itération i est toujours de $r_i * t_i$. Comme $s_i = 0$ et $r_i + t_i = x_{i-1}$, alors on en conclut que le nombre de tests d'inclusion sera maximisé si et seulement si :

$$\text{si } x_{i-1} \text{ est pair, alors } r_i = t_i = x_{i-1}/2$$

$$\text{si } x_{i-1} \text{ est impair, alors } r_i = \lfloor x_{i-1}/2 \rfloor \text{ et } t_i = \lceil x_{i-1}/2 \rceil$$

$$\text{ou } t_i = \lfloor x_{i-1}/2 \rfloor \text{ et } r_i = \lceil x_{i-1}/2 \rceil$$

On déduit de ces valeurs de r_i et t_i que :

- le nombre maximal de tests d'inclusion à l'étape 2 de l'itération i est au maximum de $x_{i-1}^2/4$,

– le nombre maximal de générations de transversaux à l'étape 3 de l'itération i est au maximum de $x_{i-1}/2 * |e_i|$ et
– le nombre maximal de transversaux minimaux à la fin de l'itération i est de $x_{i-1}/2 + x_{i-1}/2 * |e_i|$

En ne regardant que le nombre de générations de transversaux, alors on a la même remarque que pour l'agorithme 1 : le pire des cas est celui où le nombre total de transversaux minimaux est exponentiel selon la taille de l'hypergraphe et a une complexité en $\mathcal{O}(n^m)$.

Ainsi, si on arrive à construire un hypergraphe dont le nombre de transversaux minimaux augmente à chaque itération de manière exponentielle, tout en respectant les valeurs de r_i, s_i et t_i précédemment étudiées, on aura un hypergraphe qui, pour l'alogrithme 2 maximise à la fois le nombre de tests d'inclusion (et de calculs d'intersection) et le nombre de générations de transversaux minimaux. Un tel hypergraphe est présenté dans l'exemple qui suit.

Exemple 38

$$H_2 = \{\{1,2,3,4\},\{3,4,5,6\},\{5,6,7,8\},\{7,8,9,10\},\{9,10,11,12\},\{11,12,13,14\},$$
$$\{13,14,15,16\},\{15,16,17,18\},\{17,18,19,20\},\{19,20,21,22\},\{21,22,23,24\},$$
$$\{23,24,25,26\},\{25,26,27,28\}\}$$

En donnant les évolutions des nombres de 1-, n- et non-persistants et le nombre de transversaux minimaux au cours de l'exécution de l'algorithme 2, le tableau ci-dessous montre que l'hypergraphe H_2 respecte les recommandations précédentes : H_2 est bien un très mauvais cas pour l'algorithme 2 puisque le nombre de n-persistants est nul à chaque itération, les nombres de 1- et de non-persistants sont à peu près égaux à chaque itération, et le nombre de transversaux minimaux double à peu près à chaque itération (assurant un nombre total exponentiel par rapport à la taille de l'hypergraphe).

Itération	1	2	3	4	5	6	7	8	9	10	11	12	13
Nb. de 1-persistant	0	2	4	4	16	24	48	112	192	416	832	1600	3328
Nb. de n-persistant	0	0	0	0	0	0	0	0	0	0	0	0	0
Nb. de not-persistant	1	2	2	8	12	24	56	96	208	416	800	1664	3264
Nb. de tr. min.	4	6	12	28	48	104	208	400	832	1632	3264	6592	13056

○

C.4 Comparaison

Le tableau C.1 résume les différents résultats théoriques obtenus précédemment pour les très mauvais cas des algorithmes 1 et 2, et donne ainsi un moyen de comparer l'efficacité théorique réciproque et au pire des deux algorithmes.

Exemple 39

Sur les deux exemples H_1 et H_2, on obtient le tableau C.2 représenté graphiquement à la figure C.1. Ces figurent comparent l'exécution des algorithmes 1 et 2 en termes de nombre de tests d'inclusion (et calculs d'intersection), et de nombre de générations de transversaux candidats pour les cas des hypergraphes H_1 et H_2. Les valeurs sont calculées comme suit, avec i le numéro de l'itération courante, x_{i-1} le nombre de transversaux minimaux obtenus à l'itération $i-1$ et $|e_i|$ la cardinalité de l'arête examinée à l'itération i :

A l'itération i	Nb. max de générations de transversaux	Nb. max de tests d'inclusion et de calculs d'intersection
Algorithme 1	$x_{i-1} * \|e_i\|$	$(x_{i-1} * \|e_i\|) * (x_{i-1} * \|e_i\| - 1)$
Algorithme 2	$\frac{x_{i-1}}{2} * \|e_i\|$	$x_{i-1} + \frac{x_{i-1}^2}{4}$

TAB. C.1 – Valeurs maximales du nombre d'opérations élémentaires à chaque itération, pour les algorithmes 1 et 2.

– Le nombre de candidats générés par l'algorithme 1 est donné par la formule $(x_{i-1} * |e_i|)$. On rappelle que l'algorithme 2 génère directement les transversaux minimaux (et donc ne génère aucun candidat non minimal). Ce calcul se base sur celui du nombre de transversaux minimaux générés à chaque itération, valeur connue pour les deux exemples étudiés : pour H_1 le nombre de transversaux minimaux double à chaque itération, et pour H_2, il est donné dans le tableau de l'exemple 38.
– Le nombre de tests d'inclusion à l'itération i pour l'algorithme 1 est donné par la formule $(x_{i-1} * |e_i|) * (x_{i-1} * |e_i| - 1)$, puisque dans cet algorithme, on teste l'inclusion de chaque candidat avec tous les autres.
– Le nombre de tests d'inclusion à l'itération i pour l'algorithme 2 est donné par la formule $x_{i-1} + r_i * t_i$. Pour H_1, on a tout le temps $r_i = 0$ puisque les arêtes sont toutes disjointes. Ainsi le nombre de tests d'inclusion à l'itération i pour l'algorithme 2 appliqué à H_1 est de x_{i-1}. Pour H_2, les valeurs de r_i et t_i sont données dans le tableau de l'exemple 38.

Avec H_1 et H_2 qui sont de très mauvais cas pour respectivement l'algorithme 1 et l'algorithme 2, nous avons un moyen pour comparer en théorie les comportements au pire des deux algorithmes. Au vu des données du tableau C.2, et du graphique correspondant de la figure C.1, notamment concernant H_2, on peut dire que l'algorithme 2 améliore l'agorithme 1. ◦

Itération	Nb. de candidats générés par l'algorithme 1 avec H_1	Nb. tests d'inclusion dans l'algorithme 1 avec H_1	Nb. tests d'inclusion dans l'algorithme 2 avec H_1	Nb. transversaux minimaux générés avec H_1	Nb. de candidats générés par l'algorithme 1 avec H_2	Nb. tests d'inclusion dans l'algorithme 1 avec H_2	Nb. tests d'inclusion dans l'algorithme 2 avec H_2	Nb. transversaux minimaux générés avec H_2
1	2	2	1	2	4	12	4	4
2	4	12	2	4	16	240	8	6
3	8	56	4	8	24	552	14	12
4	16	240	8	16	48	2 256	44	28
5	32	992	16	32	112	12 432	220	48
6	64	4 032	32	64	192	36 672	624	104
7	128	16 256	64	128	416	172 640	2 792	208
8	256	65 280	128	256	832	691 392	10 960	400
9	512	261 632	256	512	1600	2 558 400	40 336	832
10	1024	1 047 552	512	1 024	3328	11 072 256	173 888	1 632
11	2048	4 192 256	1024	2 048	6528	42 608 256	667 232	3 264
12	4096	16 773 120	2048	4 096	13056	170 446 080	2 665 664	6 592
13	8192	67 100 672	4096	8 192	26368	695 245 056	10 869 184	13 056

Tab. C.2 – Evolution des nombres de tests d'inclusion pendant l'exécution des algorithmes 1 et 2 sur H_1 et H_2.

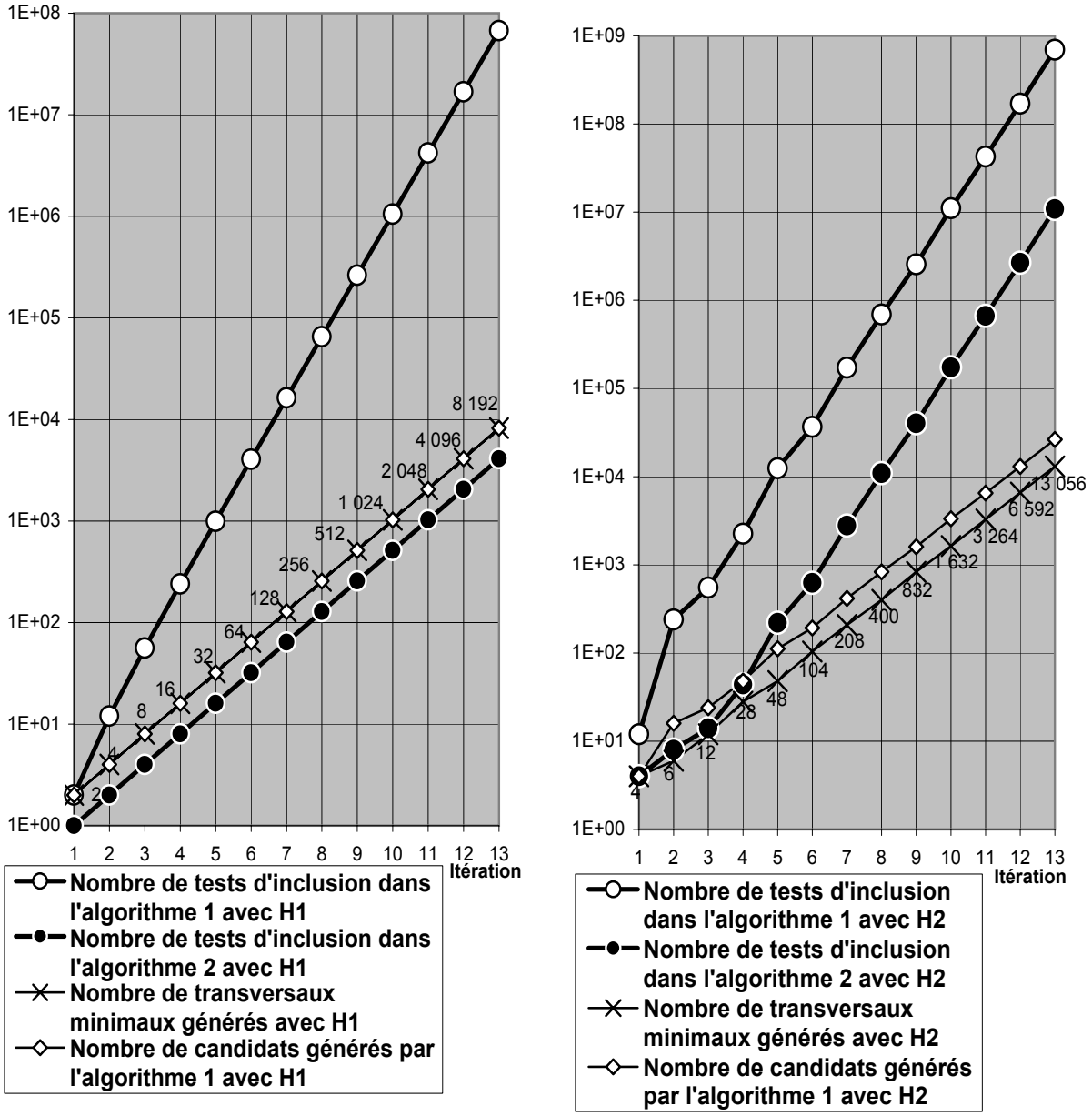


FIG. C.1 – Evolution des nombres de tests d'inclusion pendant l'exécution des algorithmes 1 et 2 sur H_1 et H_2

Annexe D

Etude détaillée des trois ontologies représentatives

On présente ici l'étude détaillée de chaque ontologie décrite dans la section 7.2.2 et de son traitement avec les 6 variantes de *computeBCov*. Pour ce faire on utilisera les représentations graphiques des mesures suivantes :

- Le temps d'exécution de chaque itération (voir la figure D.1 pour le cas 1, la figure D.4 pour le cas 2 et la figure D.8 pour le cas 3).
- Le nombre de transversaux générés (candidats et minimaux) à chaque itération (voir la figure D.2 pour le cas 1, la figure D.5 pour le cas 2 et la figure D.9 pour le cas 3).
- Le temps moyen de génération d'un transveral (candidat ou minimal) à chaque itération (voir la figure D.3 pour le cas 1, la figure D.6 pour le cas 2 et la figure D.10 pour le cas 3). Ce temps moyen est le quotient du temps total d'exécution de la phase de génération des transversaux divisé par le nombre de transversaux.

De plus, pour les cas 2 et 3, nous fournissons deux graphiques (aux figures D.7 et D.11) qui aident à voir l'effet réciproque de la génération des candidats sur le BnB et vice versa. Après l'explication de chaque cas, nous résumons les résultats relatifs à la performance de chaque phase de *computeBCov* (i.e. le BnB, la politique et les persistants) dans la figure 7.3.

Cas 1 (figures D.1, D.2 et D.3) A partir de le figure D.1, nous distinguons trois groupes de variantes de *computeBCov* :

- Les variantes très peu efficaces : celles sans BnB ni persistants pour la génération des transversaux,
- les variantes peu efficaces : sans BnB mais avec les persistants et
- les variantes efficaces : avec le BnB (le temps de chaque itération est inférieur à 100ms).

Il y a trois explications étroitement reliées qui expliquent cette répartition :

- D'abord, il est clair, à partir de la figure D.2, que le BnB limite (voire même évite dans ce cas) l'explosion combinatoire du nombre de transversaux générés qui survient quand le BnB n'est pas utilisé. Pour les variantes avec BnB, le nombre de candidats est plus ou moins constant durant les 6 itérations.
- Deuxièmement, la phase de génération (avec ou sans les persistants) est telle que le temps moyen de calcul d'un transversal augmente linéairement avec le nombre de candidats (voir les figures D.2 et D.3). La raison est simple : dans cette phase de génération, plus nombreux sont les candidats générés à l'itération précédente, plus nombreux sont ceux à générer à l'itération courante, et donc plus il est long de tester toutes les inclusions possibles afin de

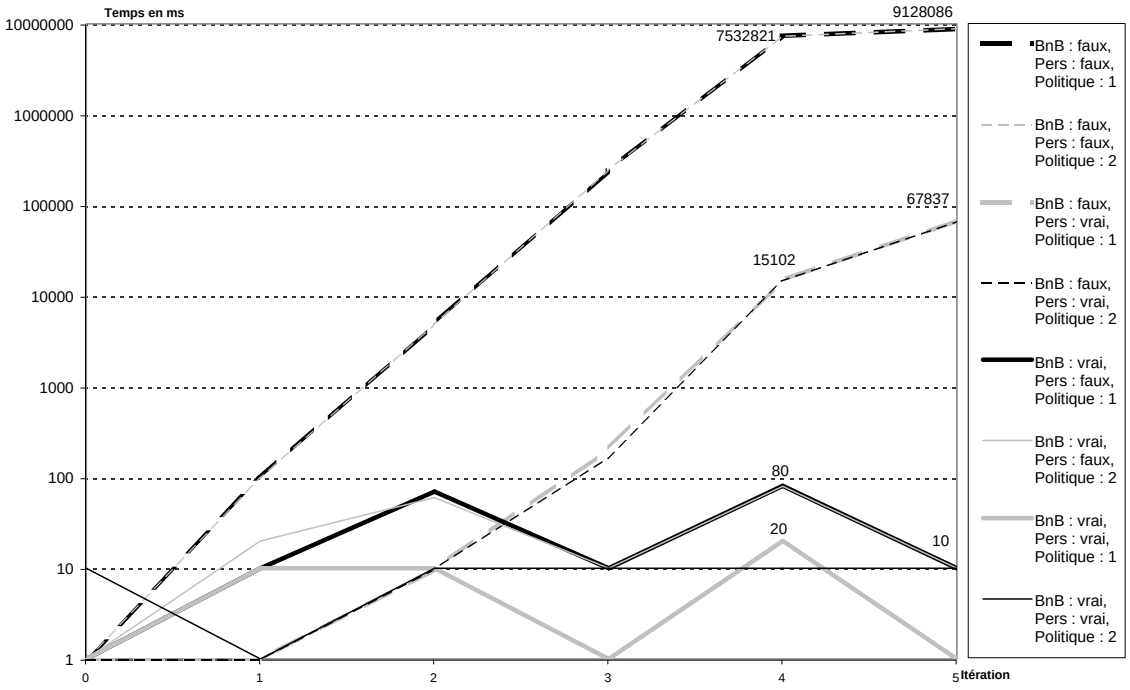


FIG. D.1 – Temps d'exécution de chaque itération de *computeBCov* durant le traitement du cas 1 par D^2CP.

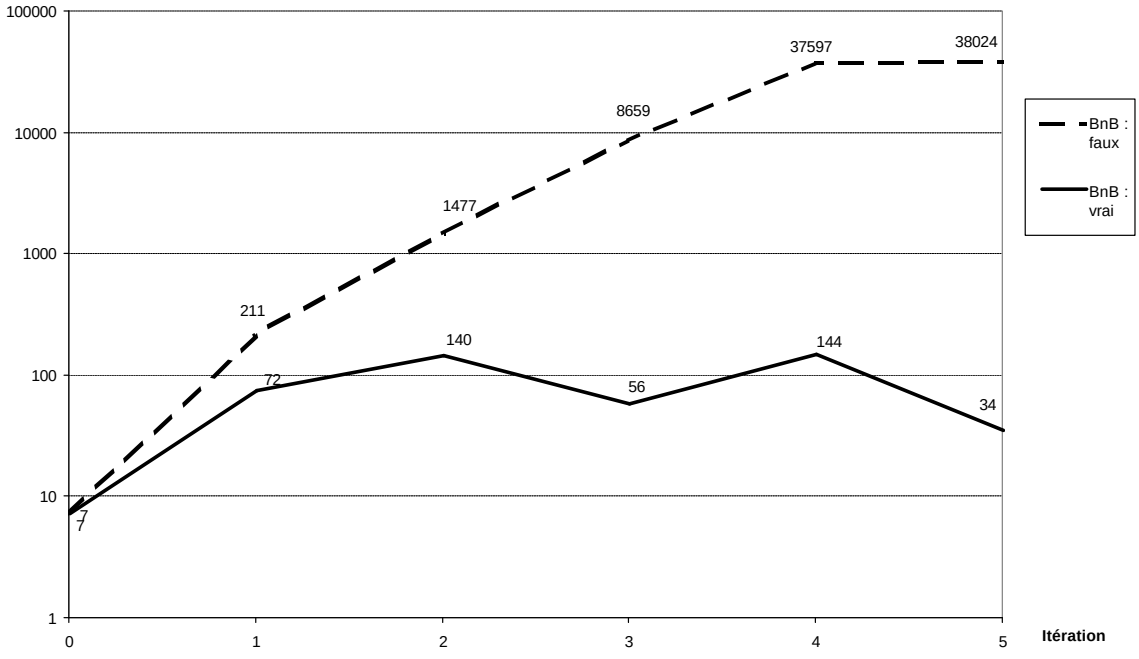


FIG. D.2 – Nombre de transversaux candidats générés à chaque itération de *computeBCov* durant le traitement du cas 1 par D^2CP.

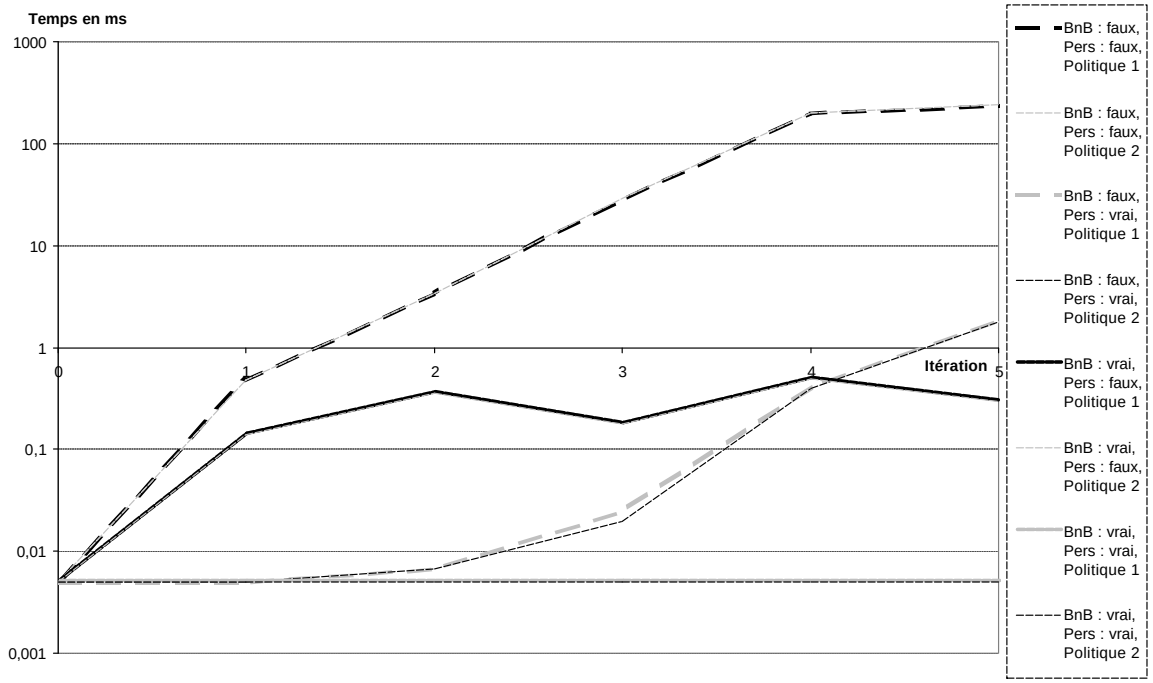


FIG. D.3 – Temps moyen de génération d'un transversal à chaque itération durant le traitement du cas 1 par D^2CP.

ne garder que les minimaux. Ainsi, quand le nombre de transversaux augmente de manière exponentielle, le temps moyen de génération d'un transversal augmente de la même nanière, ce qui implique un temps de calcul de l'itération plus long. Réciproquement, quand le nombre de candidats diminue, le temps moyen de génération d'un candidat diminue aussi.

– L'ajout des persistants dans la phase de génération apporte une nette amélioration du temps de génération moyen d'un transversal. Nous pouvons voir, à la figure D.3, que ce temps moyen évolue comme précédemment avec le nombre des transversaux, mais à un niveau bien plus bas : la tendance est la même (que sans les persistants) mais les temps sont nettement plus courts. De plus, associés avec le BnB, (i.e. quand le nombre de candidats n'explose pas), les persistants ont un effet encore plus marqué dans la réduction du temps moyen de génération d'un transversal.

En résumé :

– Le BnB limite l'explosion combinatoire du nombre de transversaux. Cela implique un temps moyen de génération plus faible.
– Les persistants (i.e. la génération des transversaux minimaux utilisant l'algorithme 2 basé sur le théorème 3.2.1) réduit beaucoup ce temps moyen de génération d'un transversal qu'il y ait ou non le BnB (si le BnB est présent, l'effet est accentué).

Cas 2 (figures D.4, D.5, D.6 et D.7) L'étude du cas 2 est limité aux variantes de *computeBCov* avec BnB, parce que les exécutions sans BnB effectuée avec D^2CP ont été arrêtées en raison d'un temps de calcul trop long (plus de 12 heures).

A partir de la figure D.4, nous pouvons faire deux remarques :

– A l'instar du cas 1, l'utilisation des persistants dans la génération implique une grande réduction du temps de génération des transversal. Ceci est très net à la figure D.7 en ayant

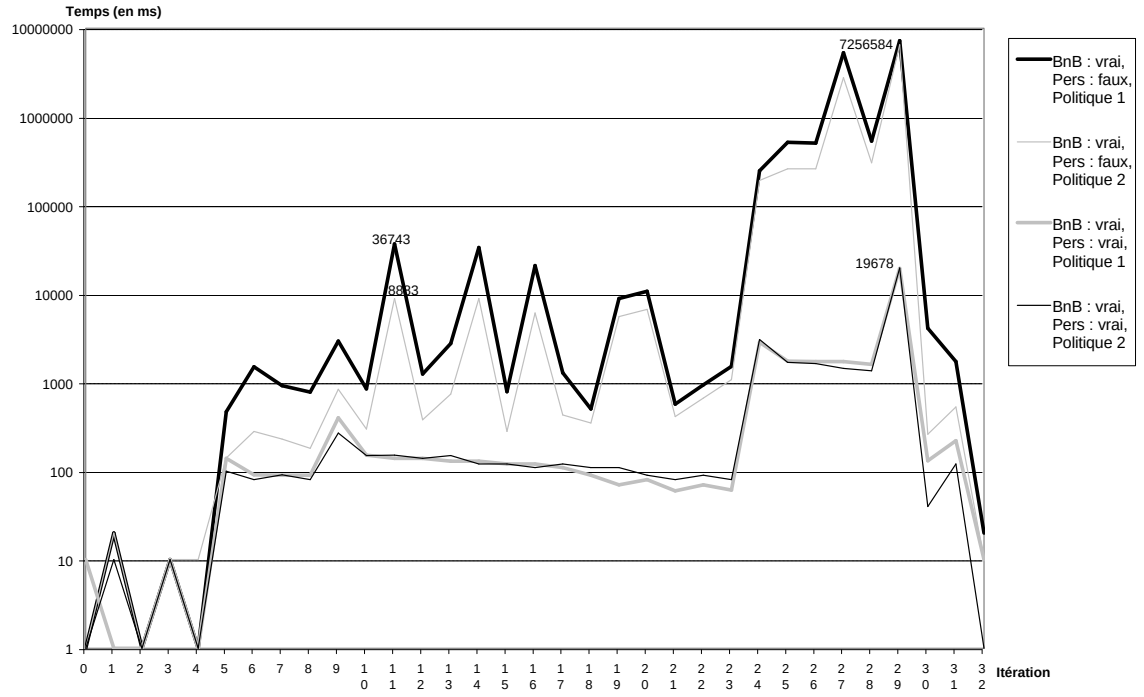


FIG. D.4 – Temps d'exécution de chaque itération de *computeBCov* durant le traitement du cas 2 par D^2CP.

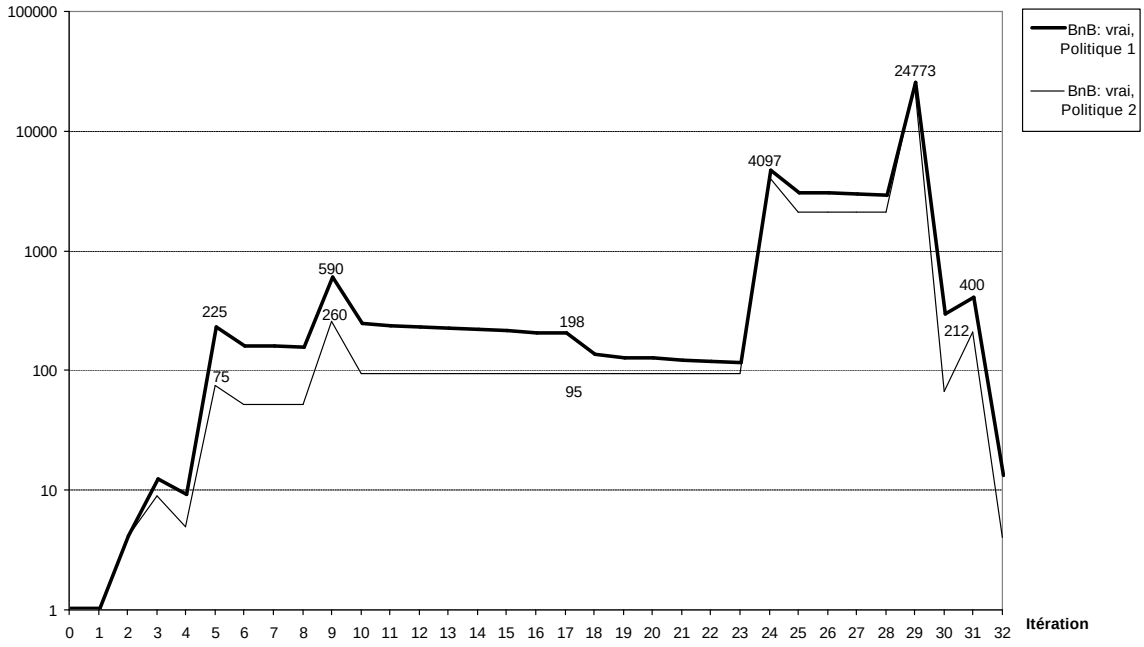


FIG. D.5 – Nombre de transversaux candidats générés à chaque itération de *computeBCov* durant le traitement du cas 2 par D^2CP.

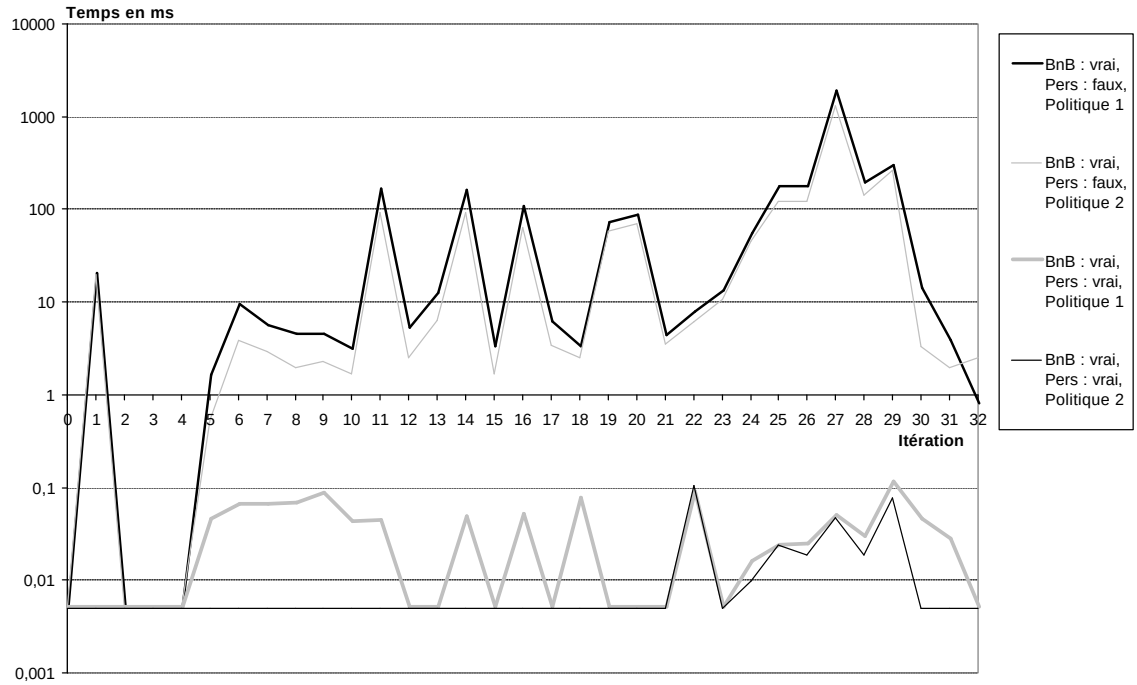


FIG. D.6 – Temps moyen de génération d'un transversal à chaque itération durant le traitement du cas 2 par D^2CP.

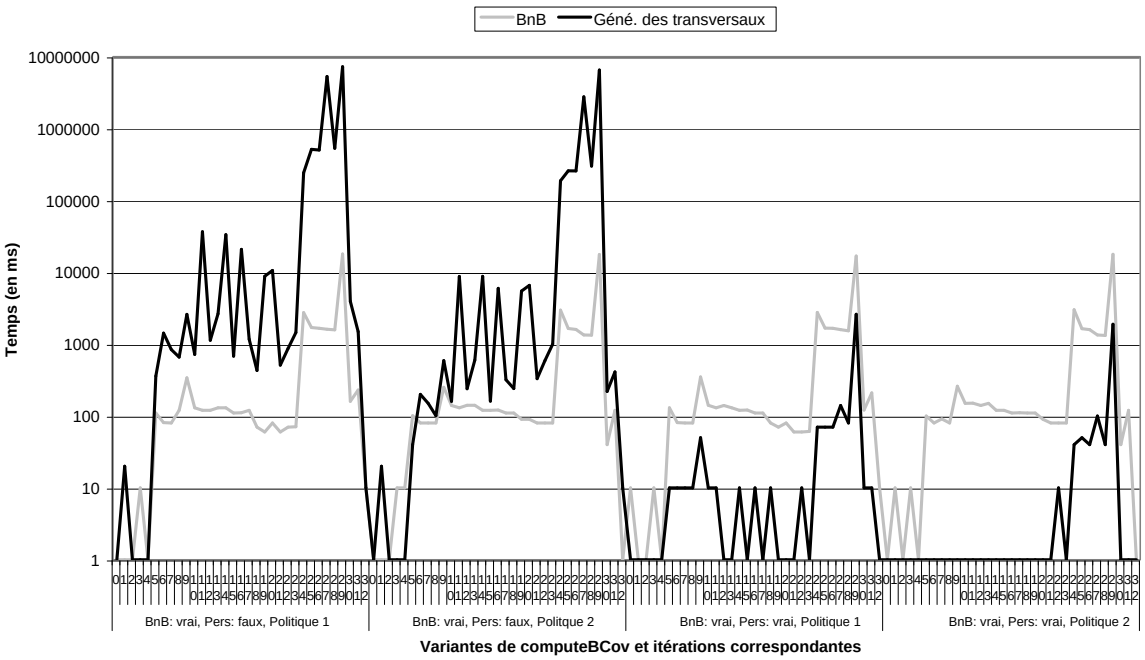


FIG. D.7 – Temps d'exécutions des étapes de BnB et de génération des transversaux à chaque itération et pour chaque variante de *computeBCov* au cours du traitement du cas 2 par D^2CP.

à l'esprit que le nombre de transversaux est à peu près le même pour les 4 variantes de *computeBCov* étudiées ici (voir la figure D.5). Comme précédemment, ceci est dû à une baisse très nette du temps moyen de génération d'un transversal (voir la figure D.6).
A côté de cela, on peut remarquer à partir de la figure D.7 que les temps des phases de BnB sont presque les mêmes pour chaque variante de *computeBCov* : comme les nombres de candidats générés sont aussi les mêmes, cela signifie que le BnB n'est pas influencé par l'usage des persistants

- La seconde remarque est que l'on peut découper le processus complet en trois phases :
  - Phase 1 : de l'itération 0 à 4, *computeBCov* démarre, et les mesures de temps ne sont pas très significatives en raison du faible nombre de transversaux générés. C'est en quelque sorte la phase de montée en charge.
  - Phase 2 : de l'itération 5 à 23, le temps de chaque itération est borné : avec les persistants, ce temps est presque constant, et sans les persistants, on peut observer des oscillations entre une borne supérieure et une borne inférieure.
  - Phase 3 : de l'itération 24 à 32, il y a une explosion dans les temps de calculs, suivie d'une chute indiquant la fin de *computeBCov*.

  Expliquons les phases 2 et 3.
  Durant la phase 2, sans les persistants, le temps de calcul de chaque itération oscille entre deux bornes alors qu'avec les persistants, ce temps est presque constant. Logiquement, la même chose se produit à la figure D.6 pour le temps moyen de génération d'un transversal. Comme dans le cas 1, cela est dû au fait qu'au cours de certaines itérations, de nombreux transversaux candidats non minimaux sont générés lorsque les persistants ne sont pas utilisés : leur génération est coûteuse en temps, de même que les tests qui déterminent qu'ils sont non minimaux. Cela confirme le résultat théorique de l'annexe C sur l'efficacité des persistants.
  Au cours de la phase 3, sans les persistants, nous pouvons observer que les itérations 27 et 29 sont très longues. La raison en est la suivante : à l'itération 27, le nombre de candidats n'est pas très élevé (moins de 4000) mais le temps moyen pour les générer est très grand (voir la figure D.6), et à l'itération 29, le temps moyen de génération n'est pas très grand, mais le nombre de candidat est très élevé (24773). Quand on utilise les persistants, comme le temps moyen de génération d'un candidat reste bas, le temps de calcul de chaque itération dépend principalement du temps de calcul réservé au BnB, et donc dépend principalement du nombre de transversaux minimaux issus de l'itération précédente.

En résumé :

- Les persistants impliquent un gain de temps très significatif dans la génération des transversaux minimaux. Il en découle que le temps de calcul d'une itération est très proche de celui du BnB de cette même itération.
- Le BnB n'est pas influencé par le temps de génération d'un transversal minimal, mais par leur nombre.

Cas 3 (figures D.8, D.9, D.10 et D.11) A partir de la figure D.8, nous continuons à observer l'efficacité des persistants. De plus, nous voyons dans ce cas l'influence de la politique choisie pour le BnB : la figure D.9 montre que la politique 2 implique une baisse du nombre de transversaux minimaux de l'itération 6 jusqu'à la fin. L'effet est le même que pour les cas 1 et 2 : moins il y a de transversaux minimaux issus de l'itération précédente, plus le temps moyen de génération des nouveaux transversaux minimaux de l'itération courante est réduit.

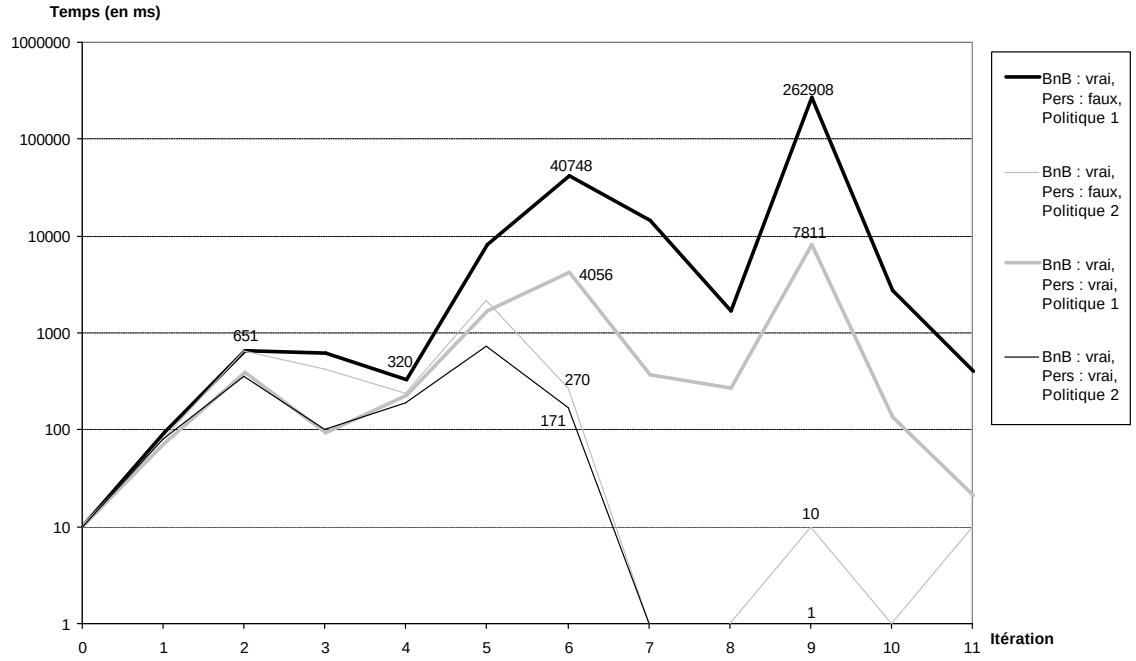


FIG. D.8 – Temps d'exécution de chaque itération de *computeBCov* durant le traitement du cas 3 par D^2CP.

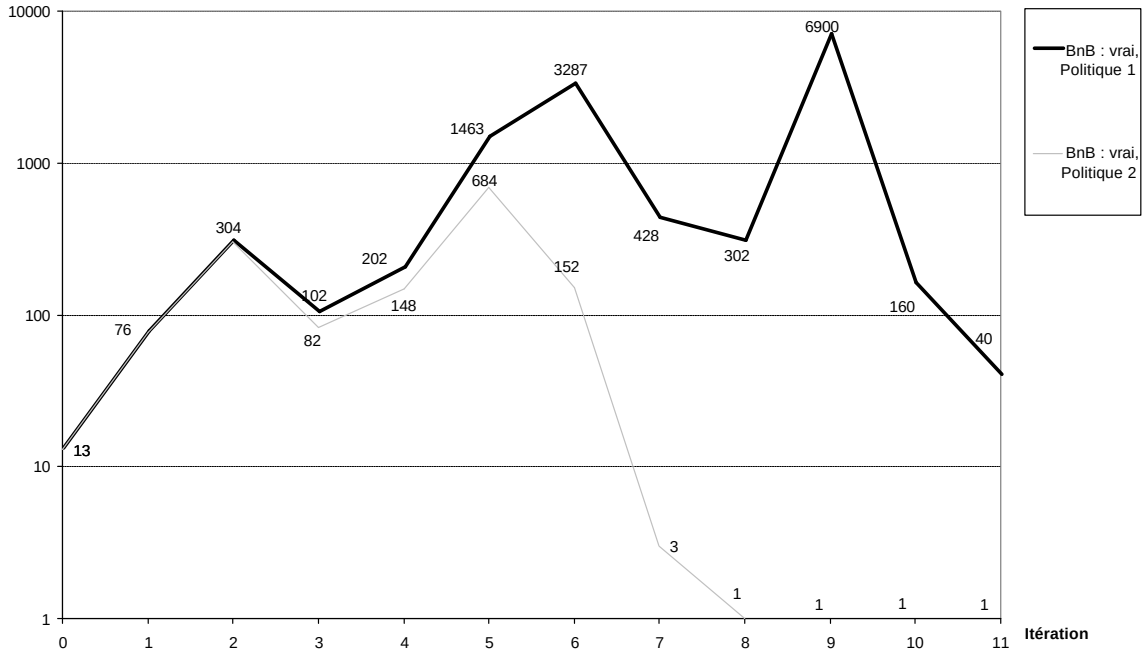


FIG. D.9 – Nombre de transversaux candidats générés à chaque itération de *computeBCov* durant le traitement du cas 3 par D^2CP.

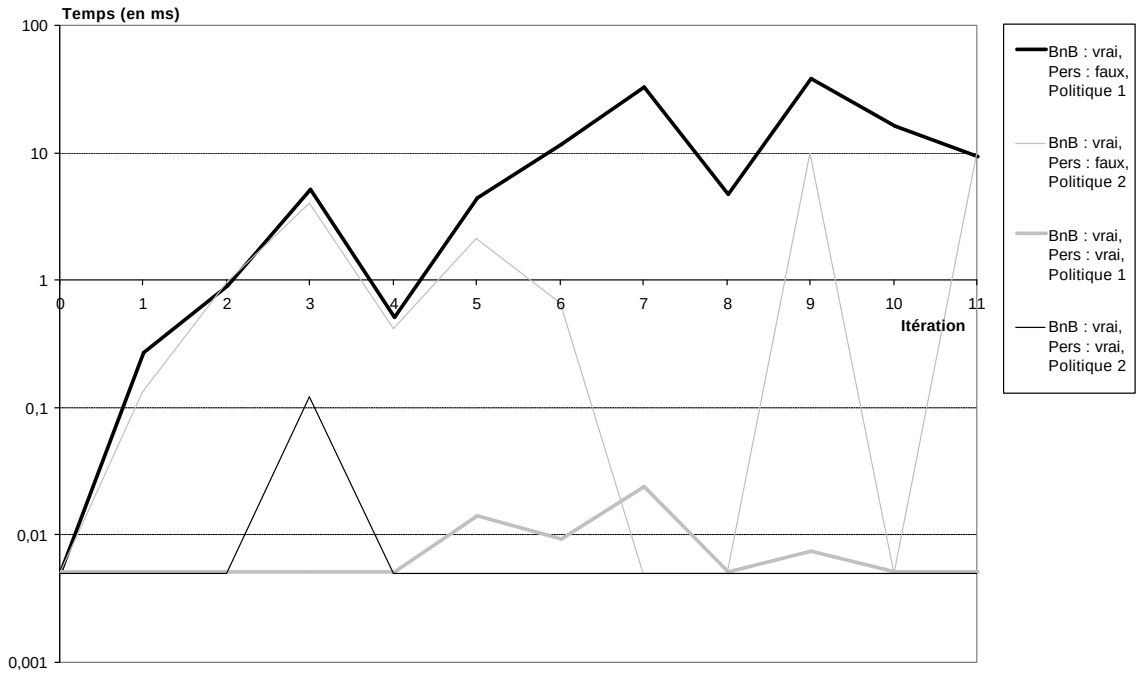


FIG. D.10 – Temps moyen de génération d'un transversal à chaque itération durant le traitement du cas 3 par D^2CP.

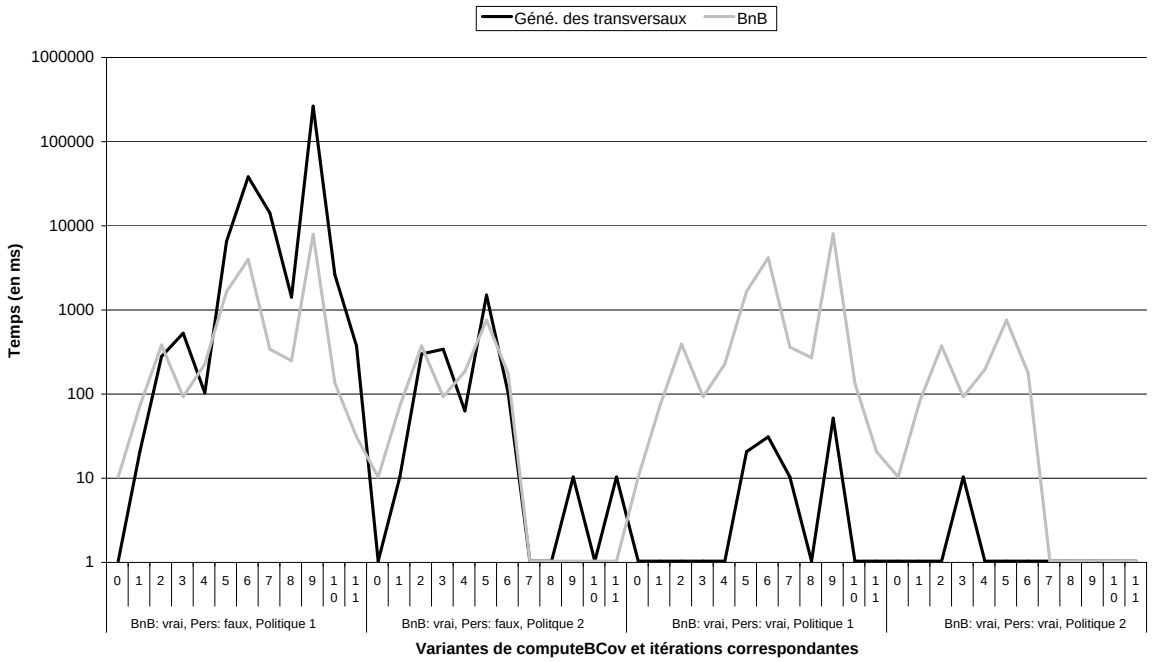


FIG. D.11 – Temps d'exécutions des étapes de BnB et de génération des transversaux à chaque itération et pour chaque variante de *computeBCov* au cours du traitement du cas 3 par D^2CP.

Nous pouvons aussi le voir à la figure D.10.

La figure D.11 résume bien ce qui se passe dans le cas 3 :

- Le BnB est amélioré par la politique 2, et, comme précédemment, il n'est pas influencé par les persistants.
- Les persistants améliorent grandement le temps consacré à la génération des transversaux minimaux.
- Comme précédemment, la variante de *computeBCov* avec le BnB, la politique 2 et les persistants est la plus rapide (quelques secondes seulement).

Annexe E

Scenarii de découverte dans le contexte de MKBEEM

Nous présentons ici quatre scenarii montrant différents aspects de la découverte de services permise par *computeBCov*. Nous donnons l'ontologie que nous avons utilisée ainsi que les requêtes qui définissent ces scenarii. L'ontologie et les requêtes sont tirées du contexte de MKBEEM. A l'adresse internet http ://www.isima.fr/~rey/demoBCover.html sont téléchargeables une courte video de démonstration des scenarii détaillés ci-dessous, ainsi que les ontologies de services et les requêtes associées sous la forme de fichiers XML.

E.1 Présentation de l'ontologie et des requêtes

L'ontologie présentée aux figures E.1 et E.2 décrit des concepts du domaine du tourisme. Elle comprend des concepts du domaine comme $Hotel$, $Camping$ ou $Voyage$, ainsi que des concepts plus généraux, comme $Date$ et $Heure$ par exemple.

Les figures E.3 et E.4 donnent l'ontologie des services. Il y a trois types de services :

- les services de voyage : par exemple $Train\_Paris$ décrit un service de voyage en train à destination de Paris nécessitant une heure et une date d'arrivée.
- les services de restauration : par exemple $Restaurant\_Japonais$ décrit un service de restauration avec des spécialités japonaises.
- les services d'hébergement : par exemple $Hotel\_Paris$ décrit un service d'hôtellerie, décrit comme un logement ayant un nombre de lits et une télévision en équipement de loisir, localisé à Paris.

La figure E.5 donne les requêtes correspondant aux quatre scénarii.

$Alpes \equiv$	$Montagne$
	$\sqcap \forall localisation\_montagne.France$
	$\sqcap \forall nom.\text{"}Alpes\text{"}$
$Appartement \equiv$	$Logement$
	$\sqcap \forall categorie.Chaine\_car$
	$\sqcap \forall nb\_chambres.Entier$
$Athenes \equiv$	$Ville$
	$\sqcap \forall localisation\_ville.Grece$
$Auberge \equiv$	$Logement$
	$\sqcap Restaurant$
	$\sqcap \forall nb\_chambres.Entier$
	$\sqcap \forall type\_pension.Chaine\_car$
$BedAndBreakfast \equiv$	$Logement$
	$\sqcap \forall equip\_loisirs.Television$
	$\sqcap \forall nb\_lits.Entier$
$Camping \equiv$	$\forall categorie.Chaine\_car$
	$\sqcap \forall type\_emplacement.Chaine\_car$
	$\sqcap Logement$
	$\sqcap \forall periode.Chaine\_car$
$Cargo\_Arrivee\_Hotel \equiv$	$Cargo\_Arrivee$
	$\sqcap Hotel$
$Couscous \equiv$	$Plat$
	$\sqcap \forall specialite.Maroc$
$Date \equiv$	$\forall jour\_sem.Chaine\_car$
	$\sqcap \forall nom\_mois.Chaine\_car$
	$\sqcap \forall num\_annee.Entier$
	$\sqcap \forall num\_jour.Entier$
	$\sqcap \forall num\_mois.Entier$
$Haute\_Montagne \equiv$	$Montagne$
	$\sqcap Haute\_Altitude$
$Haute\_Montagne\_France \equiv$	$Haute\_Montagne$
	$\sqcap \forall localisation\_montagne.France$
$Heure \equiv$	$\forall heures.Entier$
	$\sqcap \forall minutes.Entier$
$Horaire\_Arrivee \equiv$	$Voyage$
	$\sqcap \forall date\_arr.Date$
	$\sqcap \forall heure\_arr.Heure$
$Horaire\_Depart \equiv$	$Voyage$
	$\sqcap \forall date\_dep.Date$
	$\sqcap \forall heure\_dep.Heure$
$Horaire\_Depart\_Arrivee \equiv$	$Voyage$
	$\sqcap \forall date\_dep.Date$
	$\sqcap \forall date\_arr.Date$

FIG. E.1 – L'ontologie globale et du domaine (tourisme) utilisée pour illustrer le fonctionnement de *computeBCov* avec quatre scenarii. Partie 1/2.

$Hotel \equiv$	$Logement$
	$\sqcap \forall equip\_loisirs.Television$
	$\sqcap \forall nb\_lits.Entier$
$Logement \equiv$	$\forall lieu\_sejour.Chaine\_car$
	$\sqcap \forall premier\_jour.Date$
	$\sqcap \forall nb\_nuitees.Entier$
	$\sqcap \forall categorie.Chaine\_car$
$Lyon \equiv$	$Ville$
	$\sqcap \forall nom.\text{"}Lyon\text{"}$
	$\sqcap \forall localisation\_ville.France$
$Moussaka \equiv$	$Plat$
	$\sqcap \forall specialite.Grece$
$Paris \equiv$	$Ville$
	$\sqcap \forall nom.\text{"}Paris\text{"}$
	$\sqcap \forall localisation\_ville.France$
$Pizza \equiv$	$Plat$
	$\sqcap \forall specialite.Italie$
$Refuge \equiv$	$Logement$
	$\sqcap Restaurant$
	$\sqcap \forall nb\_lits.Entier$
	$\sqcap \forall type\_pension.Chaine\_car$
	$\sqcap \forall localisation\_logement.Haute\_Montagne$
$Rome \equiv$	$Ville$
	$\sqcap \forall localisation\_ville.Italie$
$Sirtaki \equiv$	$Danse$
	$\sqcap \forall specialite.Grece$
$Voyage \equiv$	$\forall lieu\_arr.Chaine\_car$
	$\sqcap \forall lieu\_dep.Chaine\_car$
	$\sqcap \forall moyen\_transport.Chaine\_car$
	$\sqcap \forall nb\_places.Entier$
	$\sqcap \forall categorie.Chaine\_car$

FIG. E.2 – L'ontologie globale et du domaine (tourisme) utilisée pour illustrer le fonctionnement de *computeBCov* avec quatre scenarii. Partie 2/2.

$Auberge\_France \equiv$	$Auberge$ $\sqcap \forall localisation\_logement.France$
$Auberge\_Paris \equiv$	$Auberge$ $\sqcap \forall localisation\_logement.Paris$
$BedAndBreakfast\_Lyon \equiv$	$BedAndBreakfast$ $\sqcap \forall localisation\_logement.Lyon$
$BedAndBreakfast\_Paris \equiv$	$BedAndBreakfast$ $\sqcap \forall localisation\_logement.Paris$
$Cargo\_Arrivee \equiv$	$Horaire\_Arrivee$ $\sqcap \forall moyen\_transport.Bateau$
$Cargo\_Depart \equiv$	$Horaire\_Depart$ $\sqcap \forall moyen\_transport.Bateau$
$Croisiere\_Arrivee \equiv$	$\forall moyen\_transport.Bateau$ $\sqcap Horaire\_Arrivee$ $\sqcap \forall categorie.Chaine\_car$
$Croisiere\_Depart \equiv$	$\forall moyen\_transport.Bateau$ $\sqcap Horaire\_Depart$ $\sqcap \forall categorie.Chaine\_car$
$Ecole\_Danse\_Grecque \equiv$	$Ecole\_Danse$ $\sqcap \forall specialite.Grece$
$Hotel\_Lyon \equiv$	$Hotel$ $\sqcap \forall localisation\_logement.Lyon$
$Hotel\_Paris \equiv$	$Hotel$ $\sqcap \forall localisation\_logement.Paris$
$Pizzeria \equiv$	$Restaurant$ $\sqcap \forall specialite.Pizza$
$Refuge\_France \equiv$	$Refuge$ $\sqcap \forall localisation\_logement.Haute\_Montagne\_France$
$Restaurant\_Grecs \equiv$	$Restaurant$ $\sqcap \forall specialite.Grece$
$Restaurant\_Indien \equiv$	$Restaurant$ $\sqcap \forall specialite.Inde$
$Restaurant\_Japonais \equiv$	$Restaurant$ $\sqcap \forall specialite.Japon$
$Restaurant\_Marocain \equiv$	$Restaurant$ $\sqcap \forall specialite.Maroc$
$Restaurant\_Marocain\_Paris \equiv$	$Restaurant\_Marocain$ $\sqcap \forall localisation\_restaurant.Paris$

FIG. E.3 – L'ontologie des services utilisée pour illustrer le fonctionnement de *computeBCov* avec quatre scenarii. Partie 1/2.

$Service\_Logement\_Precis \equiv$	$Logement$
	$\sqcap \forall equip\_loisirs.Television$
	$\sqcap \forall categorie.Chaine\_car$
	$\sqcap \forall premier\_jour.Date$
	$\sqcap \forall equip\_loisirs.Piscine$
	$\sqcap \forall equip\_loisirs.Garderie$
	$\sqcap \forall localisation\_logement.Paris$
$Service\_Logement\_Vague \equiv$	$Logement$
$Train\_Arrivee \equiv$	$Horaire\_Arrivee$
	$\sqcap \forall moyen\_transport.Train$
$Train\_Depart \equiv$	$Horaire\_Depart$
	$\sqcap \forall moyen\_transport.Train$
$Train\_Paris \equiv$	$Train\_Arrivee$
	$\sqcap \forall lieu\_arr.Paris$
$Vol\_Arrivee \equiv$	$Horaire\_Arrivee$
	$\sqcap \forall moyen\_transport.Avion$
	$\sqcap \forall nb\_places.Entier$
	$\sqcap \forall categorie.Chaine\_car$
$Vol\_Depart \equiv$	$Horaire\_Depart$
	$\sqcap \forall categorie.Chaine\_car$
	$\sqcap \forall nb\_places.Entier$
	$\sqcap \forall moyen\_transport.Avion$
$Vol\_Lyon \equiv$	$Vol\_Arrivee$
	$\sqcap \forall lieu\_arr.Lyon$
$Vol\_Paris \equiv$	$Vol\_Arrivee$
	$\sqcap \forall lieu\_arr.Paris$

FIG. E.4 – L'ontologie des services utilisée pour illustrer le fonctionnement de *computeBCov* avec quatre scenarii. Partie 2/2.

$Scenario\_1\_Flexibilite \equiv$	$Film$ $\sqcap \forall origine.Grece$ $\sqcap \forall titre.Chaine\_car$ $\sqcap \forall lieu\_arr.Paris$
$Scenario\_2\_Modelisation\_Precise\_Vague \equiv$	$Logement$ $\sqcap \forall equip\_loisirs.Television$ $\sqcap \forall localisation\_logement.Paris$ $\sqcap Restaurant$
$Scenario\_3\_Combinatoire \equiv$	$Restaurant$ $\sqcap Logement$ $\sqcap Voyage$ $\sqcap \forall fumeur.Booleen$ $\sqcap \forall equip\_loisirs.Television$
$Scenario\_4\_Inference \equiv$	$Logement$ $\sqcap \forall localisation\_logement.Alpes$

FIG. E.5 – Les requêtes représentant les quatre scenarii.

E.2 Scénario 1 : flexibilité

Le premier scenario est un exemple de découverte de services qui montre l'aspect flexible de la découverte des meilleures couvertures.

L'utilisateur veut aller à Paris voir un film grec dont il donne le titre au système qui l'interprète comme une chaîne de caractères. L'ontologie ne contient aucun service concernant le cinéma. Donc la réponse renvoyée à l'utilisateur concernera la seule information de sa requête qui correspond à certains services, à savoir le fait qu'il veut aller à Paris. Au final, il obtient deux combinaisons d'un service chacune : Vol_Paris et $Train\_Paris$.

La flexibilité de l'approche tient donc au fait qu'une information commune, même si elle ne représente qu'un faible pourcentage de la requête, peut faire découvrir des services. La conséquence est qu'une requête n'aura pas de réponse si elle est complètement en dehors du domaine de l'ontologie utilisée.

La figure E.6 montre le résultat de l'exécution de cette requête dans BCover.

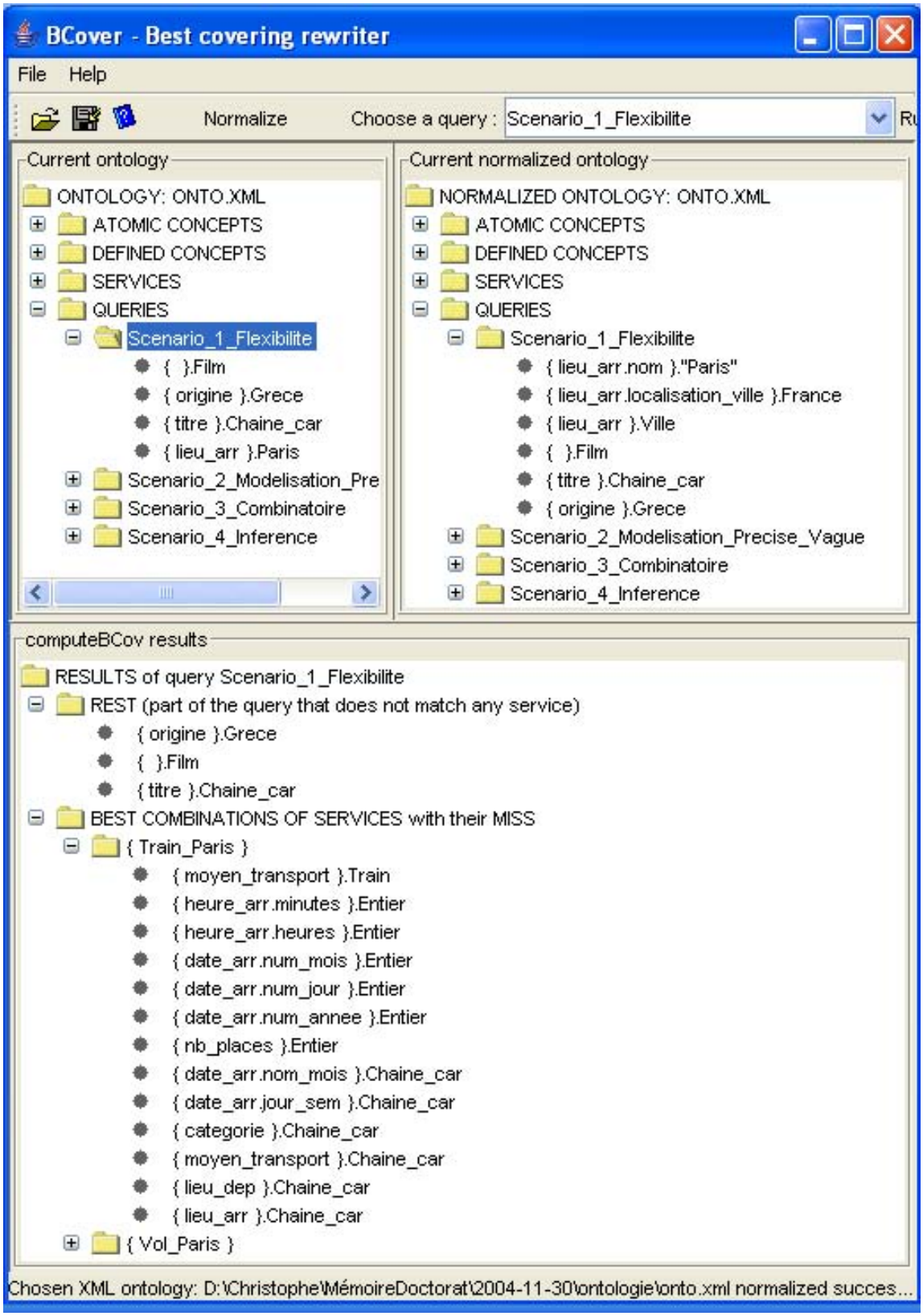


FIG. E.6 – Scénario 1 de *computeBCov* : flexibilité et interopérabilité.

E.3 Scénario 2 : modélisation vague/précise

Ce scénario montre l'importance de la minimisation des rest et miss, et comment ce critère permet à *computeBCov* de calculer des solutions pertinentes en présence de services modélisés différemment (de manière vague ou au contraire précise).

L'utilisateur cherche un restaurant et un logement équipé d'une télévision à Paris. L'ontologie contient, entre autres, quatre services de logement potentiellement intéressants qui sont :

- $BedAndBreakfast\_Paris$,
- $Hotel\_Paris$,
- $Service\_Logement\_Vague$ et
- $Service\_Logement\_Precis$.

Ces quatre services sont donnés à la figure E.7. Les deux premiers sont intéressants car ils représentent des services de logement sur Paris. Le troisième l'est car il est défini de manière très vague par la seule notion de logement sans aucune autre précision. Le dernier l'est car c'est un service de logement sur Paris défini de manière très précise avec de nombreux détails (avec télévision, piscine et garderie,...). Il s'agit ici de voir comme *computeBCov* se comporte en présence de services "normaux" (les deux premiers), vague (le troisième) et précis (le dernier). L'exécution dans BCover, montrée à la figure E.8, montre que *computeBCov* ne découvre ni $Service\_Logement\_Vague$ ni $Service\_Logement\_Precis$ en résultats de la requête. Ceci est dû au fait que $Service\_Logement\_Vague$ implique un rest non minimal, et que $Service\_Logement\_Precis$ implique un miss non minimal. On vérifie donc la pertinence des résultats de *computeBCov* induite par l'optimisation des rest et miss.

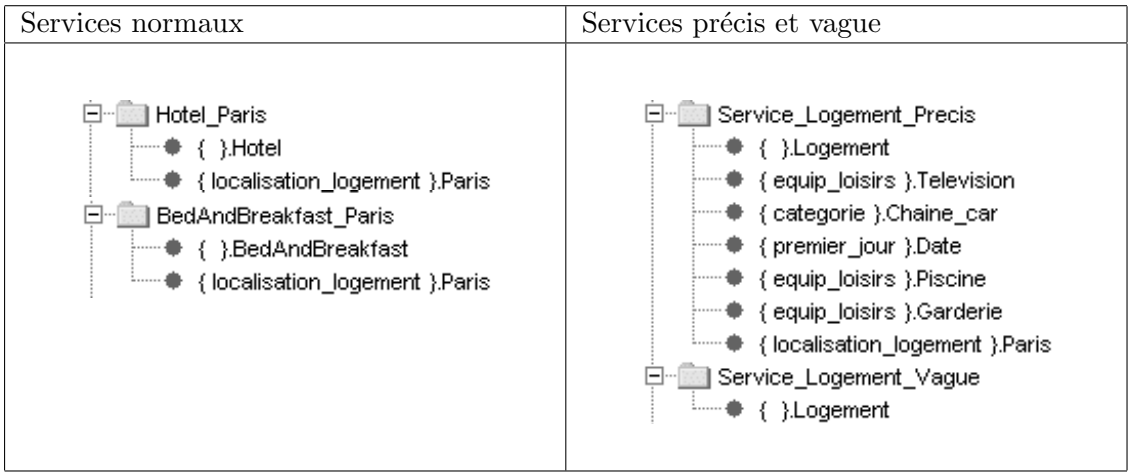


FIG. E.7 – Services utiles pour le scénario 2 de *computeBCov*.

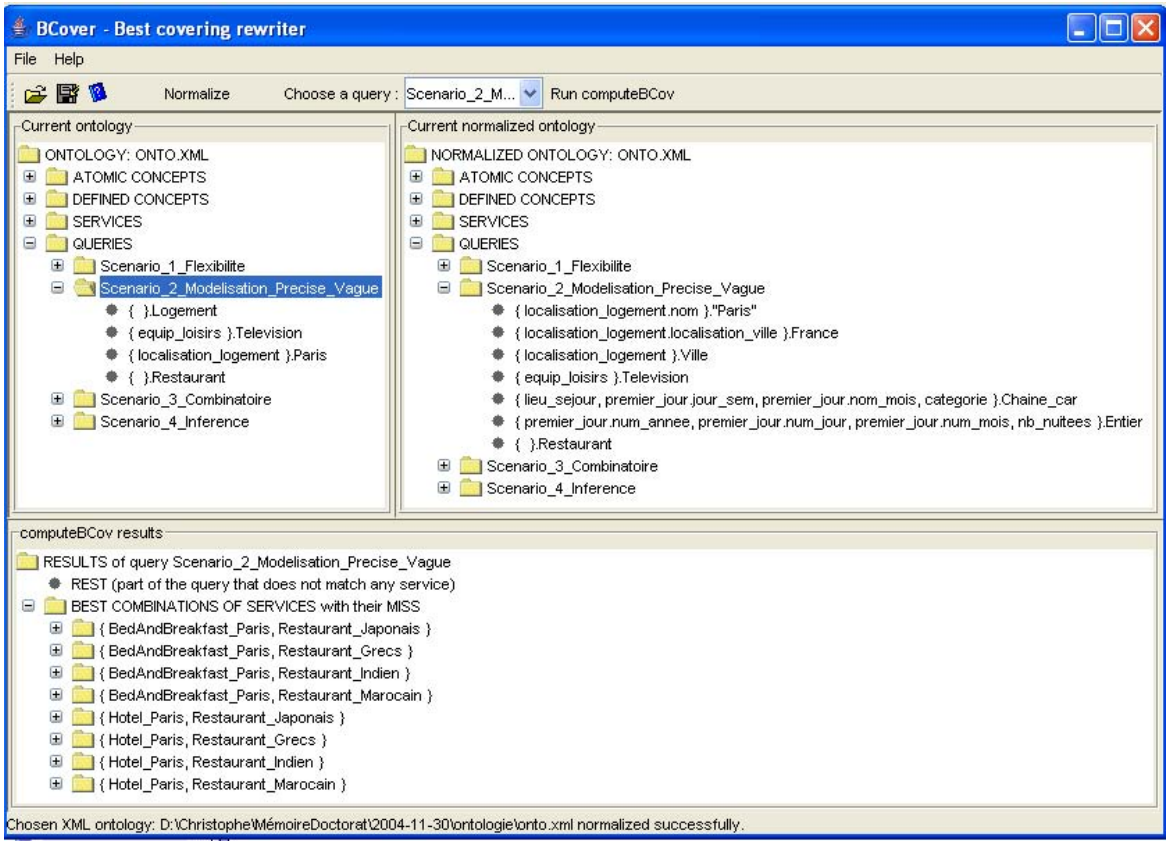


FIG. E.8 – Scénario 2 de *computeBCov* : modélisation précise ou vague des services.

E.4 Scénario 3 : combinatoire

Ce scénario montre que *computeBCov* effectue ses recherche dans un espace combinatoire et peut aboutir dans certains cas à un nombre important de résultats tous aussi bons les uns que les autres.

L'utilisateur cherche un restaurant, un logement avec télévision, un voyage et indique s'il est fumeur. C'est une requête assez vague qui aboutit à un grand nombre de solutions qui sont en fait les éléments du produit cartésien des trois ensembles suivants :

– $\{BedAndBreakfast\_Lyon, BedAndBreakfast\_Paris, Hotel\_Lyon, Hotel\_Paris\}$,
– $\{Croisiere\_Depart, Croisiere\_Arrivee, Train\_Depart, Train\_Arrivee, Cargo\_Depart, Cargo\_Arrivee, Vol\_Depart, vol\_Arrivee\}$ et
– $\{Restaurant\_Japonais, Restaurant\_Grec, Restaurant\_Indien, Restaurant\_Marocain\}$

soit 128 combinaisons de services différentes, mais toutes également pertinentes pour la requête.

La figure E.9 montre l'exécution de ce scénario par BCover.

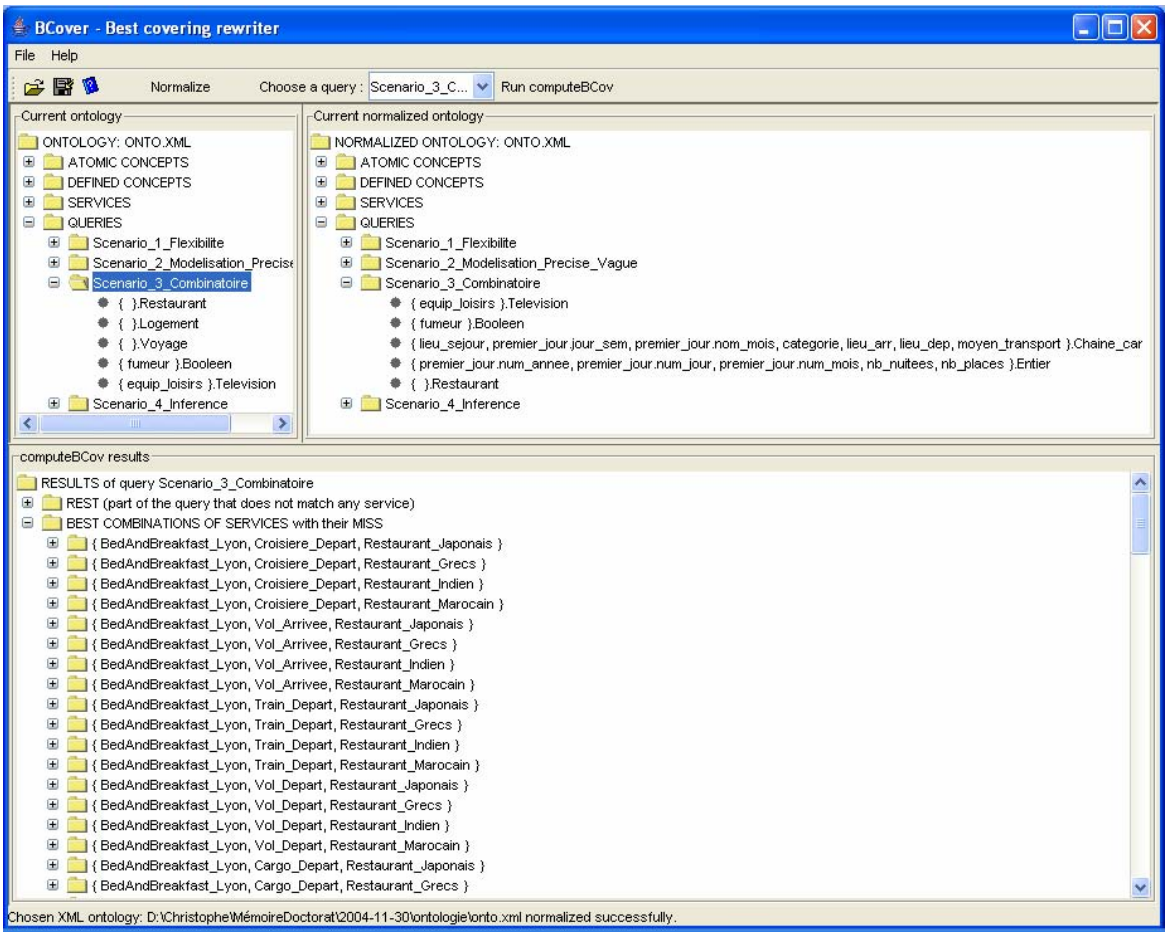


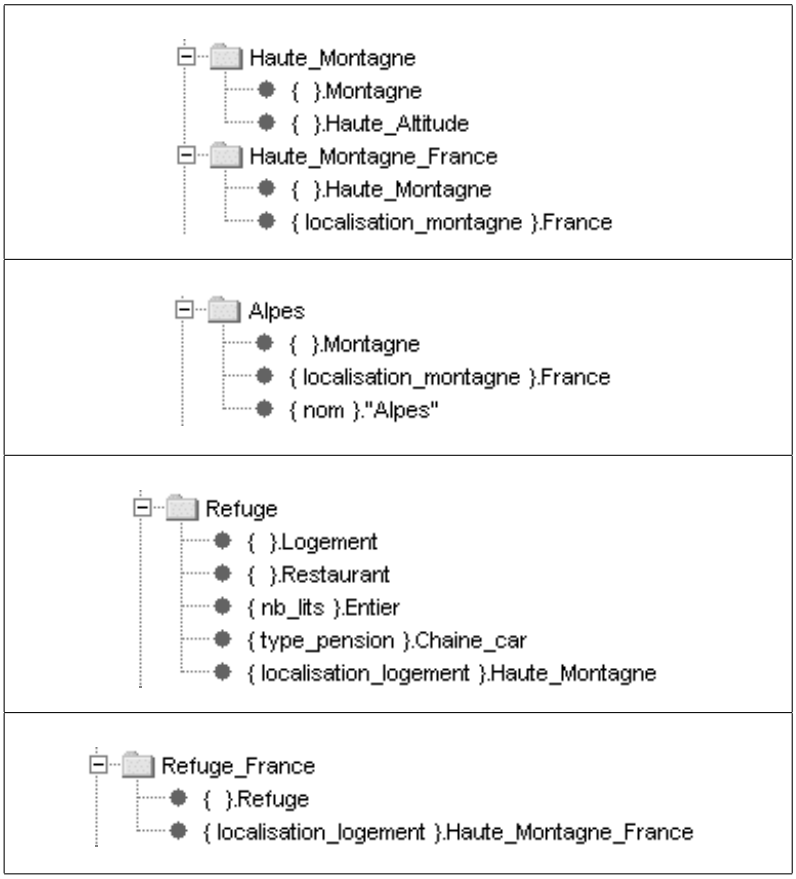
FIG. E.9 – Scénario 3 de *computeBCov* : combinatoire.

E.5 Scénario 4 : inférence

Ce scénario montre comment *computeBCov* utilise la sémantique du langage $\mathcal{FL}_0$ pour raisonner en même temps qu'il effectue sa recherche combinatoire. C'est cet aspect d'inférence logique qui permet de qualifier la découverte réalisée par *computeBCov* de sémantique (et non syntaxique).

L'utilisateur cherche un logement dans les Alpes. Il n'y a aucun service dans l'ontologie qui fournit exactement ce genre de logement. Cependant, il y a un service qui fournit des refuges en France. Par le jeu des relations de subsomption existant entre les concepts *Alpes*, $Haute\_Montagne$, $Haute\_Montagne\_France$, $Refuge$ et $Refuge\_France$, la requête obtient comme réponse le service $Refuge\_France$.

La figure E.10 montre les définitions des concepts cités et le résultat de l'exécution correspondante de *computeBCov* dans BCover.



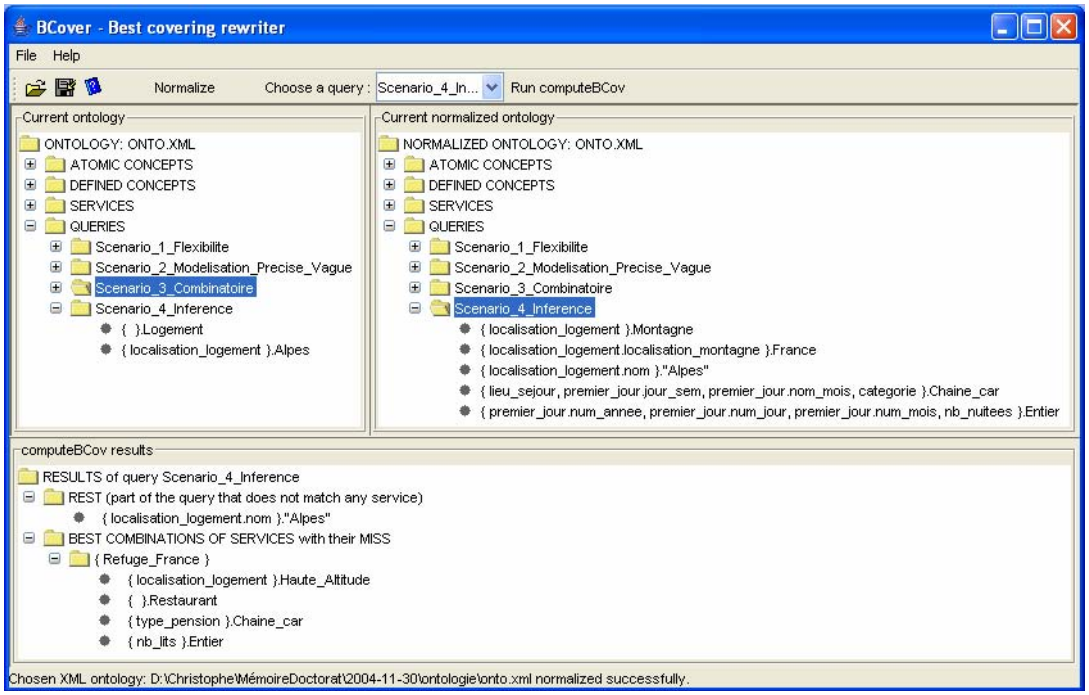


FIG. E.10 – Scénario 4 de *computeBCov* : inférence.

TITLE : Discovery of best covers of a concept using a terminology. Application to the dynamic discovery of semantic web services.

ABSTRACT : Semantic web services are a proposal for automating the management of web services, which are a new paradigm for inter-enterprises applications integration. This work is about the web services dynamic discovery problem, that is the ability to automatically find a set of web services that answer some particular query.

Throughout a theoretical framework based on description logics, we propose to formalize the problem of the dynamic discovery of web services as a new instance of concept rewriting using a terminology. We call this new instance the discovery of best covers using a terminology. It is the following problem : "given a concept Q (a query) and a terminology $\mathcal{T}$ of concepts (services), the problem amounts to find concepts conjonctions from $\mathcal{T}$ (sets of services) that are the most semantically close possible of Q". The notion of greatest semantic proximity between concepts is defined with the help of the semantic difference operator for description logics. It amounts to minimize the differences between Q and its rewritings, thus maximizing the common information between them. The rewriting criterion is interesting since it is more general thus more flexible than other rewriting criteria.

We study the discovery of best covers for the description logics having the structural subsumption property, and also for $\mathcal{ALN}$. In both cases, we show it is a NP-Hard problem that has a tight relation with the problem of computing the minimal transversals of a hypergraph. We propose two algorithms called *computeBCov* and *compute*$\mathcal{ALN}$*BCov*. A prototype implementing *computeBCov* has been developped in the context of the european project MKBEEM, which has been used to validate the approach in the tourism context.

KEY-WORDS : concept cover, best cover, rewriting, description logics, web services, semantic web services, services discovery, minimal transversals, hypergraph

TITRE : Découverte des meilleures couvertures d'un concept en utilisant une terminologie. Application à la découverte de services web sémantiques.

RESUME : Les services web dits sémantiques sont une proposition pour automatiser les tâches liées à la gestion des services web, paradigme récent pour l'intégration d'applications inter-entreprises. Cette thèse s'intéresse au problème de leur découverte dynamique, c'est-à-dire la possibilité de trouver automatiquement un ensemble de services web qui répondent à une requête, par raisonnement sur leur sémantique.

Dans le cadre des logiques de description, nous proposons de formaliser ce problème comme une nouvelle instance de réécriture de concepts en utilisant une terminologie. Cette instance est appelée découverte des meilleures couvertures et s'énonce ainsi : "étant donnés un concept Q (une requête) et une terminologie $\mathcal{T}$ de concepts (les services), le problème consiste à rechercher les conjonctions de concepts de $\mathcal{T}$ (les ensembles de services) qui sémantiquement se rapprochent le plus de Q ". La notion de proximité sémantique entre concepts est définie en s'appuyant sur l'opérateur de différence sémantique entre concepts des logiques de description. Elle consiste à minimiser les différences entre une requête Q et ses réécritures potentielles, maximisant ainsi l'information commune entre elles. Ce critère de réécriture est intéressant car plus général et donc plus souple que les relations de subsomption ou d'équivalence utilisées habituellement.

Pour les logiques de description ayant la propriété de subsomption structurelle, et pour $\mathcal{ALN}$, nous montrons que le problème est NP-Difficile et qu'il est fortement lié au problème de la recherche des transversaux minimaux d'un hypergraphe. Nous proposons deux algorithmes appelés *computeBCov* et *compute$\mathcal{ALN}$BCov*. Un prototype implémentant *computeBCov* a été développé au sein du projet européen MKBEEM, permettant la validation de l'approche dans le domaine du tourisme.

DISCIPLINE : Informatique

MOTS-CLES : couverture de concept, meilleure couverture, réécriture, logiques de description, services web, services web sémantiques, découverte de services, transversaux minimaux, hypergraphe

INTITULE ET ADRESSE DU LABORATOIRE : LIMOS, Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes. ISIMA, Campus universitaire des Cézeaux, 63177 Aubière.