Université de Montréal (Canada)
Université des Sciences et Technologies de Lille (France)

Intégration des connaissances ontologiques dans la fouille de motifs séquentiels avec application à la personnalisation Web

par
Mehdi Adda

Département d'Informatique et de Recherche Opérationnelle (Canada)
Laboratoire d'Informatique Fondamentale de Lille (France)

Thèse présentée à la Faculté des études supérieures et postdoctorales
Faculté des arts et des sciences (Canada)
et à l'école doctorale des sciences pour l'ingénieur de l'USTL (France)
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

Juillet, 2008

Université de Montréal (Canada)
Université des Sciences et Technologies de Lille (France)
Faculté des études supérieures et postdoctorales
Faculté des arts et des sciences (Canada)
et à l'école doctorale des sciences pour l'ingénieur de l'USTL (France)

Cette thèse intitulée:

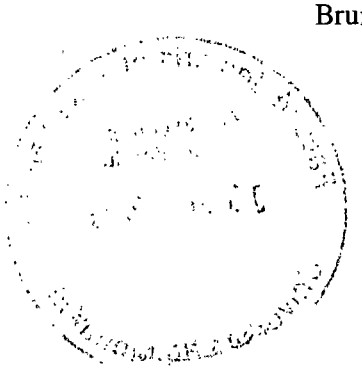
Intégration des connaissances ontologiques dans la fouille de motifs séquentiels avec application à la personnalisation Web

présentée par:

Mehdi Adda

est évaluée par un jury composé des personnes suivantes:

Gena Hahn,	président-rapporteur
Petko Valtchev,	directeur de recherche
Rokia Missaoui,	codirectrice
Chabane Djeraba,	codirecteur
Philippe Langlais,	membre du jury
Dan Simovici,	examinateur externe/rapporteur
François Lepage,	représentant du doyen de la FES
Sophie Tison,	évaluateur interne du côté français
Bruno Bachimont,	évaluateur externe (rapporteur) du côté français



Thèse acceptée le: 21 Novembre 2008

Résumé

La fouille de données vise à extraire des connaissances à partir d'un grand volume de données. Lorsque les associations et l'ordre chronologique d'apparition des items sont recherchés, les connaissances extraites sont appelées *motifs séquentiels*. Les travaux de recherche existants ont porté principalement sur l'étude de motifs séquentiels composés d'objets et dans un certain nombre de cas, de catégories d'objets (concepts). Alors que les motifs d'objets sont trop spécifiques, et de ce fait peuvent être peu fréquents, les motifs de concepts ont divers niveaux d'abstraction et risquent d'être moins précis.
La prise en compte d'une ontologie du domaine dans le processus de fouille de données permet de découvrir des motifs plus compacts et plus pertinents qu'en l'absence d'une telle source de connaissance. En outre, les objets peuvent non seulement être décrits par les concepts auxquels ils se rattachent mais aussi par les liens sémantiques qui existent entre concepts. Cependant, les approches de fouille existantes restent restrictives par rapport aux modes d'expression offerts par une ontologie.

La contribution de ce travail est de définir la syntaxe et la sémantique d'un langage de motifs qui prend en considération les connaissances incorporées dans une ontologie lors de la fouille de motifs séquentiels. Ce langage offre un ensemble de primitives pour la description et la manipulation de motifs. La méthode de fouille sous-jacente procède au parcours de l'espace de motifs par niveau en se basant sur un ensemble de primitives de navigation. Ces primitives tiennent compte de la relation de généralisation/spécialisation qui existe entre les concepts (et les relations) des motifs.

Afin de valider notre approche et analyser la performance et la mise à l'échelle de l'algorithme proposé, nous avons développé la plateforme *OntoMiner*. Tout au long de la thèse, le potentiel de notre approche de fouille a été illustré à travers un cas de recommandation Web. Il ressort que l'inclusion des concepts et des relations dans le processus de fouille permet d'avoir des motifs plus pertinents et de meilleures recommandations que les approches classiques de fouille de motifs séquentiels ou de recommandation.

Mots clés: Fouille de données, fouille du Web, motifs séquentiels, connaissances du domaine, ontologies, personnalisation.

Abstract

Data mining aims at extracting knowledge from large sets of data such as association rules, clusters and patterns. When both associations and temporal order between items are sought, the discovered knowledge are called *sequential patterns*. Existing studies were conducted mainly on sequential patterns involving objects and in some cases object categories. While patterns based on objects are too specific, non frequent patterns based on categories (concepts) may have different levels of abstraction and be possibly less precise. Taking into account a given domain ontology during a data mining process allows the discovery of more compact and relevant patterns than in case of the absence of such source of knowledge. Moreover, objects may not be only expressed by the concepts they are attached to, but also by the semantic links that hold between concepts. However, related studies that exploited domain knowledge are restrictive with regard to the expressive power offered by ontology.

Our contribution consists to define the syntax and the semantics of a pattern language which exploits knowledge embedded in an ontology during the process of mining sequential patterns. The language offers a set of primitives for pattern description and manipulation. Our data mining technique explores the pattern space level by level using a set of navigation primitives which take into account the generalization/spécialization links that hold between concepts (and relationships) contained in patterns at different abstraction levels.

In order to validate our approach and analyze the performance and scalability of the proposed algorithm, we developed the *OntoMiner* plateform.

Throughout this thesis, the potential of our mining approach was illustrated with an example of Web recommendation. We came to the conclusion that taking into account concepts and relationships of an ontology during the process of data mining allows the discovery of more relevant patterns and leads to better recommendations than those found without using background knowledge.

Keywords: data mining, Web mining, sequential patterns, domain knowledge, ontology, personalization.

Table des matières

Liste des tableaux

Liste des figures

Notation

Expression	Définition
Ω	ontologie du domaine
$\mathscr{O}_\Omega$	ensemble des objets de l'ontologie : $o_1, o_2, ..., o_l$
O^ω	ensemble de toutes les séquences d'objets pouvant être construites à partir de $\mathscr{O}_\Omega$
$\mathscr{C}_\Omega$	ensemble des noms des concepts de l'ontologie : $c_1, c_2, ..., c_m$
C^ω	ensemble de toutes les séquences de concepts pouvant être construites à partir de $\mathscr{C}_\Omega$
$\mathscr{R}_\Omega$	ensemble des noms des relations de l'ontologie : $r_1, r_2, ..., r_n$
$P_c()(resp.P_r())$	fonction qui calcule la profondeur d'un concept (resp. d'une relation)
$\sqsubseteq_\Omega$	relation de subsomption de concepts et de subsomption de relations dans Ω
$\sqsubseteq^r_\Omega$	relation de subsomption régulière de concepts ou de relations dans Ω
$dom(r)$	concepts du domaine de définition de la relation r
$ran(r)$	concepts du co-domaine de la relation r
$ran(r,c)$	co-domaine de r lorsque son domaine est restreint à c
$c_i \perp c_j$	c_i, c_j sont incomparables si c_i n'est ni parent ni enfant de c_j
$r_i \perp r_j$	les relations r_i et r_j sont incomparables si r_i n'est ni successeur ni prédécesseur de r_j
$pred_\Omega(c_i)$	ensemble des prédécesseurs du concept c_i dans Ω
ρ	ensemble de tous les triplets (concept, relation, concept) qui peuvent être construits à partir de Ω
ρ_o	ensemble de tous les triplets (objet, relation, objet) qui peuvent être construits à partir de Ω
$pred_\Omega(r_i)$	ensemble des relations plus générales que r_i dans Ω
$[]_{\mathscr{C}_\Omega}(resp.[]_{\mathscr{R}_\Omega})$	relation d'interprétation d'un concept (resp. d'une relation) dans Ω
Γ_Ω	espace de tous les motifs pouvant être construits à partir de Ω
Δ	ensemble de séquences d'objets
Δ_Ω	ensemble de séquences d'objets étendues

Tableau 1 – Notations utilisées dans le cadre de cette thèse.

Expression	Définition
$E(X)$	ensemble des parties de l'ensemble X
$\emptyset_{\Gamma_\Omega}$	motif vide
$S.\varsigma[i]$	concept de la position i du motif S
$\sqsubseteq_{\Gamma_\Omega}$	relation de subsomption de motifs dans Γ_Ω
$\tau()$	fonction qui calcule le rang d'un motif
$[]_{\Gamma_\Omega}$	relation d'interprétation d'un motif
$\lhd$	relation d'instanciation d'un motif par une séquence d'objets étendue
$\rhd$	relation de couverture partielle d'une séquence d'objets par un motif
$\psi_c^1()$	fonction qui calcule l'ensemble critique d'un concept racine
$\psi_c^n()$	fonction qui calcule l'ensemble critique d'un concept non racine
$\psi_r^1()$	fonction qui calcule l'ensemble critique d'une relation racine
$\psi_r^n()$	fonction qui calcule l'ensemble critique d'une relation non racine

Tableau 2 – Notations utilisées dans le cadre de cette thèse (suite).

Je tiens à rendre hommage à toute ma famille, et plus particulièrement à mes très chers parents, ma femme et à mes frères et soeurs qui m'ont apporté un soutien à toute occasion, et qui m'ont accordé tout au long de mes études une entière attention.

Mes pensées les plus profondes s'adressent aussi à mes belles soeurs et beaux frères et à mes nièces et neveux.

Cette thèse leur est entièrement dédiée.

Remerciements

Je tiens tout d'abord à remercier Mme. Sophie Tison, Mr. Bruno Bachimont, Mr. Philippe Langlais et Mr. Dan Simovici d'avoir accepté de rapporter/évaluer ce manuscrit et pour leurs remarques constructives. Toute ma gratitude s'adresse à Mr. Gena Hahn pour avoir accepté de présider le jury de cette thèse.

Je remercie tout particulièrement Mme. Rokia Missaoui, Mr. Petko Valtchev et Mr. Chabane Djeraba pour m'avoir accueillie au sein de leurs équipes respectives et pour m'avoir encadré et soutenu tout au long de cette thèse. Je les remercie également pour tout ce qu'ils m'ont appris et pour leur grande disponiblité pour répondre à mes questions. La pertinence de leurs réponses et de leurs questions m'ont beaucoup appris sur la conduite de travaux de recherches de façon générale et en particulier sur mon propre sujet de recherche. Ils sont à mes yeux des personnages plus qu'attachant, auprès desquels j'ai énormément appris.

Finalement, je tiens à remercier de tout coeur mes parents, mes frères et mes soeurs qui m'ont encouragé et soutenu dans les moments difficiles.

Je tient à souligner que sans l'apport financier du *Patrimoine canadien* (Fonds des réseaux de recherche sur les nouveaux médias) et de *Valorisation Recherche Québec* (VRQ) (Projets structurants) à travers le consortium de recherche universitaire *CoRIMedia*, ce travail n'aurait pas vu le jour.

Chapitre 1

Introduction

Ce travail s'inscrit dans le cadre de la thématique de fouille de motifs séquentiels et leur application à des fins de recommandation de contenus Web. L'approche théorique proposée est fondée sur l'élaboration d'algorithmes de fouille de données et l'exploitation des éléments (concepts et relations) d'une ontologie du domaine.
Les deux principaux axes de recherche concernés sont donc l'ingénierie des ontologies pour une représentation riche et structurée des connaissances du domaine et la fouille de données qui fournit les outils nécessaires pour l'analyse des données sur les comportements des utilisateurs.

1.1 Problématique

1.1.1 Utilisateurs avec des préférences différentes accédant un contenu diversifié

De par la nature du Web, de grandes quantités de données sont accessibles par des utilisateurs qui, d'un côté, ont des appartenances géographiques, religieuses et culturelles variées et, d'un autre côté, ont des motivations et des objectifs différents. Qu'il s'agisse de sites de commerce électronique, de services Web ou de systèmes d'accès à divers types de données (documents, musique, vidéo, images) destinés à différentes catégories d'usage, les utilisateurs se soucient de la personnalisation de leur contenu. Le point commun entre ces sites, systèmes et services est de chercher à présenter l'information la plus pertinente sous une forme qui répond le mieux aux attentes des utilisateurs. À cette fin, ces systèmes et services évaluent l'adéquation du contenu pour un utilisateur ou un groupe d'utilisateurs afin d'assurer l'accès aux ressources pertinentes seulement

ainsi que la navigation dans un grand espace de ressources. Cette personnalisation peut être vue comme la mise en correspondance entre l'expression explicite et/ou implicite des besoins d'un utilisateur et du contenu offert par un système Web [1].

À titre d'illustration, supposons qu'un utilisateur consulte un site du tourisme électronique (*e-tourisme*) contenant plusieurs milliers d'objets (destinations, activités de loisirs, infrastructures d'accueil, ...). Le système doit être en mesure de suggérer automatiquement à cet utilisateur le contenu le plus pertinent et le plus proche des ses préférences et attentes.

De manière plus concrète, si un utilisateur navigue dans le site aux pages Web de *New York* et de *Paris*, on aimerait savoir ce que le système aura à proposer à cet utilisateur compte tenu, d'une part, de l'historique des visites de cet utilisateur ainsi que d'autres utilisateurs du système, et d'autre part des descriptions détaillées des objets de contenu du système.

1.1.2 Pertinence d'un contenu pour un utilisateur

Le problème principal qui se pose dans le domaine de la personnalisation du contenu du Web est lié aux capacités des systèmes et approches proposés à évaluer l'adéquation et la pertinence d'un contenu pour un utilisateur ou groupe d'utilisateurs [8]. Pour résoudre ce problème, différentes approches sont utilisées. Parmi les approches les plus communes, on cite celles exploitant des techniques de filtrage du contenu et celles basées sur la fouille de l'usage du Web (ou *WUM* pour *Web Usage Mining*). Ces dernières exploitent les informations statiques collectées sur les utilisateurs ou détectent des régularités dans les données d'usage du système afin de construire des profils utilisateurs.

Dans les approches qui recherchent les régularités dans l'usage, les données concernant les utilisateurs sont souvent collectées de manière implicite (fichiers *logs*, par exemple) et peuvent être utilisées pour la construction des profils dynamiques des utilisateurs. D'autres données peuvent aussi être collectées explicitement (formulaires et questionnaires, par exemple) pour donner lieu aux profils statiques des utilisateurs. L'analyse de

1. Nous allons utiliser le l'expression *système Web* pour faire référence à toute application, service, site dont le contenu est accessible via le Web.

ces données mène à l'extraction de connaissances exploitables à des fins de *recommandation*. La recommandation Web est définie comme étant la prédiction de la pertinence des objets de contenu d'un système pour un utilisateur ou groupe d'utilisateurs et de suggérer les objets ayant les plus grandes prédictions [83]. Dans le cadre de cette thèse, on s'intéresse particulièrement à l'usage du Web, c'est-à-dire aux profils dynamiques des utilisateurs.

Les techniques de personnalisation Web peuvent être classées en deux catégories suivant l'approche utilisée pour mesurer le degré de pertinence d'un objet de contenu pour un utilisateur donné. D'un côté, on a les approches basées sur la mesure de la similarités numérique entre les descriptions des objets et les descriptions des utilisateurs. Cette catégorie d'approches est communément connue sous le nom de *filtrage d'information* et regroupe le filtrage orienté contenu, le filtrage orienté utilisateur, aussi appelé filtrage collaboratif, ainsi que le filtrage hybride. D'un autre côté, on a les approches qui se basent sur la détection de régularités dans les données d'usage.

Il convient de signaler que différentes formules sont utilisées pour mesurer la similarité existante entre les objets de contenu (cas du filtrage orienté contenu), ou pour retrouver les utilisateurs ayant un comportement et des préférences similaires (cas du filtrage collaboratif). Parmi ces mesures, nous citons quelques formules aussi utilisées pour la fouille de texte, et le repérage de l'information qui sont : *TF/IDF* (*cf.* section 2.2.4 du chapitre 2), la corrélation du *cosinus* et de la fonction de *Pearson*.

Dans le filtrage orienté contenu, on cherche à mesurer le degré de similarité d'un objet de contenu par rapport à un autre pour ensuite suggérer à un utilisateur les objets similaires à ceux qu'il a déjà visités. Cette catégorie d'approches ne prend en considération que les descriptions des objets de contenu du système Web et par conséquent les caractéristiques statiques et dynamiques des utilisateurs ne sont pas considérées. Ainsi, les mêmes objets peuvent être recommandés à des utilisateurs ayant des préférences différentes. Une autre limitation du filtrage orienté contenu réside dans la difficulté de recommander à un nouvel utilisateur des objets de contenu étant donné qu'il n'a pas d'historique de visites. Ce problème est couramment appelé problème du nouvel utilisateur.

Le filtrage orienté utilisateur (filtrage collaboratif ou *FC*), quant à lui, s'intéresse à mesurer la similarité existante entre les différents utilisateurs et ainsi recommander à un utilisateur les objets qui sont visités par d'autres utilisateurs qui ont les mêmes préférences. Contrairement au filtrage orienté contenu, le filtrage collaboratif va recommander des objets différents à deux utilisateurs ayant des profils différents. Le problème du nouvel utilisateur est moins crucial avec ces approches qui peuvent se contenter de la partie statique des profils des utilisateurs contrairement au filtrage orienté contenu qui exige un historique de visite pour chaque utilisateur. En effet, le *FC* permet de considérer une mesure de similarité entre utilisateurs qui se base seulement sur les profils statiques pour résoudre le problème.

Cependant, un nouveau problème est soulevé dans les approches basées sur le *FC*. Il s'agit du problème du nouvel objet. En effet, un nouvel objet qui n'est encore consulté par aucun utilisateur ne peut être recommandé, alors qu'avec le filtrage orienté contenu, un nouvel objet sera recommandé aux utilisateurs ayant visité des objets similaires à celui-ci.

Comme nous venons de le voir, chacune des ces deux approches a ses avantages et ses inconvénients. Afin de tirer profit des avantages de ces approches, une nouvelle approche de filtrage de contenu est proposée. Il s'agit du filtrage hybride qui combine les deux approches pour une recommandation plus efficace que lors de l'utilisation d'une seule catégorie d'approche.

Le grand problème des approches de filtrage d'information réside dans la difficulté du passage à l'échelle car le contenu des sites Web est de plus en plus volumineux. Il en est de même pour le nombre d'utilisateurs. À titre d'exemple, les utilisateurs du site Amazon [2] se comptent par millions et le nombre de pages du site se compte par milliers. Pour remédier à ce problème, on ne garde que les informations les plus pertinentes sur les utilisateurs et sur leur utilisation du site, et c'est dans cette optique que les techniques du *WUM* sont utilisées.

Parmi les techniques du WUM les plus répandues, nous citons celles qui exploitent le regroupement et la classification d'objets ou d'utilisateurs et les règles d'association.

2. Amazon : www.amazom.com.

L'extraction de règles d'association est une des techniques de la fouille de données les plus utilisées pour la personnalisation Web. Une règle d'association se présente sous la forme $A \rightarrow B$ ou A et B sont deux groupes d'objets de contenu (produits, pages visitées, ...) appelés motifs. Le groupe d'objets A est appelé la prémisse et le groupe d'objets B est appelé la conclusion. Une règle d'association nous renseigne sur la dépendance existante entre les éléments de la prémisse et les éléments de la conclusion par rapport à un ensemble de données de départ.

Étant donné que les règles d'association classiques ne prennent pas en considération l'ordre chronologique d'apparition des différents objets, une autre catégorie d'approches est utilisée : il s'agit de rechercher les motifs séquentiels où l'aspect temporel est considéré.

Étant donné une base de données, un motif consiste en un ensemble de propriétés (ex. produits) qui se manifestent, directement ou indirectement, au moins une fois dans un enregistrement de la base. Un motif séquentiel est un motif dont l'ordre d'apparition des éléments dans le temps est pris en considération. Il s'agit de manifestation directe lorsque les éléments du motif sont puisés de la base de données, et de manifestation indirecte dans le cas où les éléments du motif ne sont pas explicitement présents dans la base mais liés à des éléments existants dans la même base. Intuitivement, plus les descriptions des données sont riches et précises, plus est grande la possibilité de pouvoir retrouver des représentations abstraites plus précises de ces données dans les motifs.

1.1.3 Représentation de plus en plus riche des connaissances du domaine

Avec l'avènement des standards de description du contenu, en particulier les technologies autour du futur Web sémantique [23], une plus grande quantité d'informations à propos des consultations d'un utilisateur devient disponible. En conséquence, plus que de simples pages, il s'agit pour l'analyse du comportement de l'utilisateur d'examiner des séquences d'objets "visités", par exemple, à des fins d'achat, de visualisation, d'écoute, d'envoi de données ou de requêtes, etc. D'un côté, ces informations sont sou-

vent représentées en utilisant des structures de données complexes telles que les arbres et les graphes. Ainsi, le problème de prise en considération de ces structures de données dans le processus de fouille est soulevé. La solution préconisée consiste à développer des techniques de fouille de motifs à structures complexes telles que la fouille de graphes et d'arbres fréquents.

D'un autre côté, les descriptions correspondant aux données collectées sont de plus en plus organisées sous forme d'une ontologie, en particulier dans le domaine du commerce électronique où l'interopérabilité des applications des divers partenaires commerciaux est nécessaire et d'une grande actualité. Actuellement, les connaissances d'un domaine sont représentées de manière plus riche et plus structurée. Ainsi, les objets d'un domaine sont représentés non seulement avec leurs attributs et leurs classes d'appartenance mais aussi par leurs liens avec les autres objets du domaine. Ceci est rendu possible par les derniers développements sur les ontologies, particulièrement OWL qui est un standard du W3C (*cf.* chapitre 4). Le nouveau défi est de concevoir des techniques de fouille qui prennent en considération ces connaissances à différents niveaux d'abstraction. Une première tentative d'intégration des connaissances du domaine dans le processus de fouille a donné lieu à une nouvelle catégorie de motifs. Il s'agit des motifs généralisés où les catégories d'objets à différents niveaux d'abstraction sont considérés. Si nous considérons l'exemple d'une base de deux séquences $\langle Grenoble, Hotel\_Splendid \rangle$ et $\langle Montreal, Le\_St-James \rangle$ avec $Hotel\_Splendid$ et $Le\_St-James$ deux hotels, la séquence suivante $\langle City, Hotel \rangle$ est un motif séquentiel généralisé. En effet, les objets *Grenoble* et *Montreal* sont des instances de *City* et $Hotel\_Splendid$ et $Le\_St-James$ sont instances de $Hotel$.

Cependant, d'autres connaissances du domaine, telles que les liens inter-objets, restent inexploitées. Nous abordons cet aspect en détail par la suite.

1.2 Hypothèses

L'hypothèse principale de ce travail repose sur l'impact de l'exploitation des connaissances du domaine dans le processus de personnalisation Web. Ces connaissances vont

des catégories d'objets aux liens inter-objets et leur abstraction en relations inter-catégories. En effet, nous considérons que les objets/produits sélectionnés par un utilisateur sont porteurs de connaissances à différents niveaux d'abstraction et que la prise en considération de ces différents niveaux rendra le système de personnalisation plus efficace.

Ainsi, nous supposons que le parcours des objets de contenu suit les liens sémantiques du domaine. Ces liens peuvent être présents dans la représentation des connaissances du domaine de manière directe ou non (succession de plusieurs liens sémantiques). En effet, nous considérons que les liens sémantiques inter-objets sont un élément révélateur du comportement d'un usager, et que très souvent le passage entre deux objets de contenu, c'est à dire la visite par l'utilisateur d'un, puis de l'autre, suit un tel lien entre les deux. Ceci est vrai dans des contextes où la recommandation a démontré son utilité, tels les systèmes tutoriels intelligents et les applications de vente en ligne [14, 87].

Après le survol des travaux reliés à notre problématique, nous estimons que le comportement dynamique d'un utilisateur (profil dynamique) peut être représenté par trois niveaux d'abstraction suivant le degré de richesse de la représentation des connaissances utilisées. Le premier niveau est **le niveau instance** où la représentation des objets de contenu se limite à de simples identifiants. À ce niveau, les profils dynamiques seront composés des objets les plus fréquents au sein des données d'usage. Le deuxième niveau est **le niveau concept** où des concepts à différents niveaux d'abstraction sont utilisés pour représenter les objets. Chaque concept représente un ensemble d'objets ayant des caractéristiques communes. Le comportement dynamique est représenté par les concepts et suit l'intuition suivante. Si on remplace dans les données d'usage les objets par les concepts qui les représentent, un profil dynamique sera alors composé des concepts les plus fréquents au sein de cette nouvelle représentation des données. Le troisième et dernier niveau est **le niveau relation** où les différents objets de contenu sont représentés non seulement par les catégories d'objets mais aussi par les liens sémantiques entre ces objets. Ainsi, si on substitue les objets par des concepts et qu'on ajoute les relations existantes entre les concepts, nous allons obtenir une structure de graphe et les profils dynamiques seront représentés par les sous-graphes les plus fréquents qui seront ainsi composés de concepts et des relations qui les lient. Se contenter d'un seul niveau (niveau

instance ou niveau concept) ne permet l'extraction que d'une partie de ces connaissances et ne reflète que partiellement le comportement de l'utilisateur.

Nous estimons que pour obtenir une représentation plus précise du comportement dynamique des utilisateurs, et par conséquent aboutir à une personnalisation plus efficace, l'ensemble des connaissances du domaine disponibles sur les objets de contenu devraient être exploitées dans le processus du WUM. Motivé par ces observations, nous étudions les problèmes posés par la prise en compte des liens inter-objets dans l'extraction et l'application de motifs comportementaux. Dans l'état actuel de la représentation des connaissances, l'une des technologies les plus avancées est celle des ontologies. En effet, ces dernières constituent un support intéressant pour la représentation des différents types de données et métadonnées relatives à un domaine.

Afin de pouvoir intégrer les connaissances qu'offre une ontologie dans un système de personnalisation, nous proposons de considérer les motifs composés d'entités d'une ontologie comme les instances, les classes et les relations [1, 3–5]. En d'autres termes, au lieu de se contenter de la co-occurrence d'instances (d'objets) lors de la recherche de motifs, notre système s'intéresse à la co-occurrence d'entités de différents niveaux d'abstraction (instances, classes et relations entre classes). Cette logique permet de révéler des comportements qui ne sont pas visibles au niveau instance, et devrait permettre ainsi de mettre en évidence une nouvelle catégorie de motifs qui ne sont pas détectables par les méthodes existantes. Dans le cadre de ce présent travail, nous supposons donc l'existence préalable d'une ontologie liée au domaine d'application du système considéré.

Au niveau conceptuel d'un système Web, les objets manipulés peuvent avoir différents niveaux de granularité. Ainsi, un objet peut être simple ou composé. Dans le premier cas, une référence dans l'ontologie du domaine est disponible pour chaque classe d'objet. Dans le deuxième cas, on peut avoir des références à des objets composants sans y trouver de référence pour l'objet composite. Par exemple, un objet image d'un paysage comporte plusieurs objets correspondant à des entités du monde réel (ciel, nuages, arbres, soleil, etc.) et qui peuvent être représentées par des concepts au sein d'une ontologie. La manipulation de l'image en tant que telle revient à la manipulation des objets la consti-

tuant avec les relations qui les lient (proximité, distance, etc.). Face à la complexité liée à l'imbrication des objets et concepts d'une ontologie, nous nous limitons aux objets simples.

1.3 Raisonnement et motivation

Les approches de recommandation existantes effectuent des prédictions en se basant soit sur des motifs d'objets individuels ou sur des motifs de catégories d'objets. Dans les approches exploitant les objets individuels, un utilisateur ayant visité la page Web de l'application représentant la ville de *New York* se voit proposer les éléments du site qui sont souvent consultés par les mêmes utilisateurs ayant consulté cette page. Cette catégorie d'approches garantit une grande précision quant à la représentation des données de départ pour la simple raison qu'à partir d'un motif, et que si on garde la même base de données de départ, on ne risque de retrouver que les données qui sont à l'origine de son extraction.

Étant donné que les éléments d'un ensemble X sont souvent accédés conjointement, avec les approches qui exploitent les catégories du domaine, on suggère à l'utilisateur ayant visité l'objet *New York* (où $NewYork \in X$) les objets du site appartenant aux concepts de l'ensemble $X - \{NewYork\}$. Les objets à suggérer peuvent ne pas avoir été visités mais ils appartiennent à des catégories dont au moins quelques instances ont été déjà visitées. Un motif de catégories peut couvrir plus de données que celles qui ont permis de le générer. En d'autres mots, des instances d'un motif de catégories peuvent ne pas figurer dans l'historique des consultations des utilisateurs. Par exemple, si à partir d'une base de données d'usage les objets *NewYork* et $TheBrooklynMuseumofArt$ sont assez fréquents, le motif de concept composé de *City* et *Museum* auquel ils donnent lieu, peut être instantié non seulement en le couple *NewYork* et $TheBrooklynMuseumofArt$ mais aussi en tous les couples d'objets du site qui appartiennent respectivement aux concepts *City* et *Museum*. Par conséquent, un motif de catégories, dont les éléments de base sont de niveau d'abstraction élevé, a une grande portée comparé aux motifs individuels mais en même temps risque d'induire à une imprécision.

Il est à noter qu'actuellement seules les connaissances de niveau instances (objets de contenu) et de niveau concepts (catégories d'objets de contenu) sont exploitées. D'autres connaissances du domaine ne sont pas considérées dans le processus de fouille de motifs pour la personnalisation Web. En effet, des aspects importants de la description des objets, en particulier les liens sémantiques entre les objets et les relations entre les différentes catégories qui généralisent les liens et qui sont des éléments d'une ontologie, sont ignorés.

Avec la disponibilité de ces connaissances, on se pose la question de savoir quels sont les éléments d'une ontologie qui seront bénéfiques pour la recommandation et par conséquent comment les incorporer dans le processus de fouille de motifs résumant le comportement dynamique des utilisateurs. Notre préoccupation principale est d'exploiter les connaissances du domaine disponibles pour extraire des motifs qui ont une grande précision et qui ont un bon niveau d'abstraction.

1.4 Objectifs

L'objectif principal de notre travail consiste à développer une méthodologie pour la personnalisation Web qui exploite l'historique de consultation des utilisateurs ainsi que le maximum de connaissances du domaine qui sont disponibles. Afin de trouver un compromis entre précision des motifs et leur niveau de généralité (abstraction), nous avons l'idée de combiner les connaissances de niveau instances, classes et relations pour caractériser le comportement des utilisateurs de sites ou de services Web. Cette combinaison est rendue possible grâce aux connaissances offertes par une ontologie du domaine.

Dans cette méthodologie, le comportement dynamique des utilisateurs sera représenté par des motifs séquentiels. Ces derniers intégreront les connaissances ontologiques disponibles, à savoir les liens inter-objets, les concepts du domaine pour avoir un bon niveau d'abstraction et les relations inter-concepts pour augmenter la précision. Ainsi, nous avons défini un cadre de fouille de motifs qui prend en considération les connaissances ontologiques du domaine. Il s'agit d'intégrer les catégories d'objets (concepts), les relations inter-concepts ainsi que les liens inter-objets dans le processus de fouille de

motifs séquentiels qui seront désignés par le terme *motifs de concepts/relations*. C'est ainsi que nous avons développé un langage pour la représentation des motifs ainsi que des algorithmes de fouille pour explorer l'espace des motifs. Afin de mener des expérimentations et de pouvoir manipuler les différentes entités produites par un tel langage, une plateforme dédiée est construite.

1.5 Méthodologie

Pour atteindre les objectifs précédemment cités et mettre en place les fondements de la nouvelle méthodologie, une démarche de travail nous est nécessaire. Les grandes lignes d'une telle démarche se résument aux points suivants : proposition d'un langage de motifs, problèmes découlant du langage de motifs à étudier et validation de l'approche de fouille. Ces points sont détaillés dans les sous-sections qui suivent.

1.5.1 Langage de motif proposé

L'analyse et le traitement des données collectées (sessions) a comme objectif d'extraire des motifs de comportement (*web usage mining*) où deux critères principaux sont à considérer : le type de motifs à extraire et les connaissances à prendre en considération.

1. **Connaissances exploitées :** selon le type de connaissances exploitées, nous définissons deux catégories de motifs : (i) *motifs individuels* dont les constituants sont les objets (instances) de contenu et les liens entre ces objets et (ii) *motifs généralisés* qui sont constitués de classes et de relations d'une ontologie du domaine.
2. **Type de motifs recherchés :** du point de vue données traitées, nous distinguons deux types de motifs où l'ordre chronologique est respecté. (i) Les *motifs standards* où les séquences sont composées de suites d'objets. (ii) Les *motifs de concepts / relations* qui sont des séquences de concepts et de relations entre ces concepts et qui sont puisés de l'ontologie du domaine.

Il est à noter que les motifs standards sont des motifs individuels et que les motifs de concepts/relations sont des motifs généralisés.

En se basant sur cette décomposition, nous avons observé ce qui suit.

- un motif généralisé résume un motif individuel en offrant un niveau d'abstraction plus élevé qui augmente la portée des motifs généralisés avec le risque de diminuer leur précision,
- un motif de concepts/relations est la généralisation d'un motif standard qui augmente sa portée par l'utilisation de concepts à la place d'objets et qui introduit une imprécision. En effet, lors de "l'instanciation" d'un motif d'objets, un objet ne peut être remplacé que par un objet identique alors que lors de l'instanciation d'un motif de concepts, un concept peut être remplacé par tout objet lui appartenant.
- Grâce aux relations inter-concepts, la précision quant aux données et connaissances représentées est plus grande dans les motifs relationnels que dans le cas des motifs de concepts/relations. En effet, l'ajout de relations aux concepts des motifs réduit le nombre de couples d'objets qui peuvent être instanciés et de ce fait permet d'augmenter la précision en ne considérant que les couples potentiellement proches des couples d'objets ayant permis de générer ce motif. En d'autres mots, les motifs de concepts/relations préservent les liens existants entre les objets, et ainsi lors de l'instanciation on ne considère que les couples d'objets qui sont liés par les liens de même type que les relations inter-concepts du motif.

Pour illustrer ces observations, nous considérons le cas d'un motif constitué des objets de contenu *New York* et *The Brooklyn Museum of Art*. Un motif de catégories d'objets associé à ce motif serait composé des deux concepts *City* et *Museum*, et ce motif couvre non seulement le motif d'objets de départ mais aussi tout couple d'objets dont le premier est une instance de *City* et le deuxième est une instance de *Museum*. Cependant, il y a une information qui est perdue lors du passage du motif d'objets au motif de concepts. En effet, l'information qui consiste à dire que le musée *The Brooklyn Museum of Art* se trouve dans la ville de *New York* n'est plus disponible au niveau du motif de concept associé. Dans le cas où ce motif est utilisé pour générer des recommandations, on aura suggéré à un utilisateur ayant visité des objets de contenu de type *City* tous les objets de contenu de type *Museum* sans distinction entre ceux se trouvant dans cette ville et le reste des objets de type *Museum* présents dans le système. Il est à remarquer que cette

distinction est facilement faite si on avait matérialisé le lien sémantique existant entre *New York* et *The Brooklyn Museum of Art* lors du passage aux catégories de concepts. Un tel lien peut être puisé dans l'ontologie du domaine et se traduit par la relation *has-Museum* entre les deux concepts *City* et *Museum*. Avec un nouveau motif contenant la relation *hasMuseum*, seulement les musées qui ont une relation d'appartenance avec la ville visitée seront suggérés. Par cette simple illustration, nous confirmons l'observation citée ci-haut quant à la précision apportée par l'incorporation des liens inter-concepts aux motifs de concepts/relations.

Les approches de fouille que nous proposons consistent à rechercher des motifs qui prennent en considération les liens inter-objets, les concepts et les relations inter-concepts ainsi que l'ordre de parcours des objets de contenu.

1.5.2 Problèmes à étudier

Au niveau de l'extraction des motifs, la prise en considération de toutes les connaissances du domaine nous amène à l'étude des motifs ayant une structure complexe. La nature des graphes que nous manipulons est de loin plus complexe que celle des graphes simples. En effet, les motifs de concepts/relations présentent non seulement une structure de graphes mais sont aussi composés de noeuds et d'arcs à différents niveaux d'abstraction. Par conséquent, la syntaxe et la sémantique de ces motifs sont à étudier de manière rigoureuse et une formalisation adéquate doit être proposée. Par syntaxe et sémantique, nous faisons référence à la syntaxe et la sémantique des données de départ et des motifs de concepts/relations qui en sont extraits.

Étant donné que les motifs ciblés sont des motifs qui prennent en considération l'ordre chronologique de parcours des concepts, les motifs séquentiels sont au coeur de notre problématique. L'intégration de ces motifs au sein d'un système de personnalisation suppose l'existence d'un mécanisme qui permet le passage d'un motif à des suggestions. Ceci peut être réalisé en recherchant des règles d'association qui respectent la contrainte de précédence (ordre de parcours) entre objets à partir des motifs retrouvés. Une fois que le cadre général de cette nouvelle catégorie de motifs est défini, des

techniques de fouille adaptées à ces motifs sont à développer. Il s'agit essentiellement de développer des méthodes de parcours de l'espace de ces motifs à la recherche des motifs les plus intéressants (dans notre cas les plus fréquents).

L'autre problème qui risque de se poser concerne le nombre élevé de motifs qui peuvent être générés par la présence d'un grand nombre d'utilisateurs et d'un contenu riche et varié. Le nombre de motifs extraits peut être réduit si on ne cherche que les motifs les plus importants, qui en d'autres mots sont suffisants pour représenter l'ensemble de tous les motifs. Dans la littérature, on distingue principalement deux : les *motifs fermés* et les *motifs maximaux*. Cet aspect n'est pas traité dans cette thèse et est laissé pour des développement futurs.

1.5.3 Validation

La validation de notre travail se subdivise en deux phases. Durant la première phase, notre tâche consiste à développer et à mettre en place un environnement pour la manipulation des structures de données résultantes. L'outil à développer servira aussi bien pour la validation de l'approche que pour faciliter son intégration dans des environnements réels à des fins de recommandation. Un tel système sera chargé d'extraire les motifs des données collectées du site et offrir des outils de manipulation et de visualisation des motifs.

La deuxième phase de validation consiste à utiliser l'outil développé pour mener une étude expérimentale consistant à tester la mise à l'échelle des algorithmes d'extraction proposés. En d'autres mots, il s'agit de s'assurer que les algorithmes de fouille s'exécutent sur différents jeux de données.

1.6 Contribution

Dans la présente thèse, nos principales contributions sont comme suit :

- Proposition et formalisation d'un langage de description de motifs pour l'intégration des connaissances du domaine dans le processus de fouille sous forme de

motifs de concepts/relations ;
- Développement d'une méthodologie et de deux algorithmes de fouille de motifs de cette nouvelle catégorie ;
- Extension de la plateforme *Protégé* pour rechercher les motifs de concepts/relations sous forme d'un plugiciel (*plug-in*) ;
- Exploitation des motifs de concepts/relations pour la personnalisation Web.

1.7 Plan de la thèse

Notre document est organisé comme suit : le chapitre 2 présente une évolution des techniques de personnalisation Web, allant des approches classiques aux approches qui intègrent les connaissances du domaine avec les avantages et les inconvénients inhérents à ces approches. De plus, nous abordons dans ce chapitre les questions ouvertes posées dans le domaine de la personnalisation Web, questions qui sont à la base du travail de recherche que nous avons effectué. Le chapitre 3 est un survol de la représentation des connaissances d'un domaine à l'aide des ontologies et le formalisme des logiques de description.

Les chapitres 4 et 5 sont consacrés à la présentation de notre approche de fouille de motifs. Alors que le chapitre 4 est dédié à la présentation du langage de fouille des motifs, le chapitre 5 est consacré à la présentation de la solution algorithmique pour la fouille des motifs décrits par le langage proposé. Afin de montrer le potentiel de la nouvelle catégorie de motifs, nous présentons dans le chapitre 6 l'exploitation des motifs proposés à des fins de recommandation Web. Le chapitre 7 est consacré à la présentation de l'implémentation du cadre de fouille proposé, et c'est dans ce même chapitre que nous rapportons les résultats des expérimentations effectuées. Finalement, une conclusion générale de cette thèse ainsi que des perspectives de travaux futurs sont présentées dans le chapitre 8.

Chapitre 2

Personnalisation Web

Ce chapitre représente l'état de l'art sur l'évolution de la personnalisation Web qui couvre aussi bien des approches classiques que des approches faisant appel aux technologies du Web sémantique. Nous allons aussi aborder les questions ouvertes posées dans le domaine de la personnalisation Web, questions qui sont à la base du travail de recherche que nous avons effectué.

2.1 Évolution de la personnalisation Web

Le point commun entre les systèmes et les sites qui se soucient de l'adaptation et de le personnalisation du contenu est de présenter l'information la plus pertinente sous une forme qui répond le mieux aux attentes des usagers. Selon [62], un site personnalisé est un site qui reconnaît les utilisateurs, qui collecte des informations sur leurs préférences et adapte les services offerts dans le bon format et au bon moment. Le degré de pertinence de l'information est relatif. Il dépend à la fois du contexte du client (langue, situation géographique, etc.) et de ses intentions. La personnalisation Web peut être répartie en trois grandes catégories :

- Individualisation (*Customization*) : une personnalisation contrôlée par l'utilisateur. L'utilisateur choisit l'information qui lui sera délivrée et comment elle le sera. Ainsi, le système ne fait que répondre aux exigences explicitement formulées par l'utilisateur (ex. personnaliser la page d'accueil individuelle sur un portail Web : couleurs, taille de la fonte, dispositions des objets, etc.).
- Recommandation : une personnalisation contrôlée par le système. Contrairement à la *customization*, la recommandation tend à répondre aux requêtes d'un utilisateur en tenant compte, entre autres, de ses précédentes expériences de navigation

ou de l'expérience d'autre utilisateurs. À titre d'exemple, le système construit un modèle représentant les besoins individuels de chaque utilisateur en se basant sur les données d'usage, et fournit un contenu suivant ce modèle.

- Adaptation : mettre le contenu sous un format qui est le plus susceptible d'être adapté au contexte de l'utilisateur et de son environnement. Cela reviendrait par exemple à adapter le format de la vidéo au client matériel utilisé (cellulaire, ordinateur de poche, ordinateur portatif,...), et prendre en considération des contraintes liées aux supports de délivrance des données.

Comme nous venons de le voir, la principale différence entre la *customization* et la recommandation consiste à savoir à qui on donne le contrôle de l'expérience de navigation Web. Dans le cas de la *customization*, l'utilisateur a le plein contrôle alors que dans le cas de la recommandation, c'est le système qui en a le contrôle et c'est lui qui décide du contenu à servir et guide l'interaction avec l'utilisateur suivant l'expérience passée et présente de l'utilisateur ainsi que de ses préférences explicitement fournies.

Dans le cadre de cette thèse, nous nous intéressons davantage aux techniques de recommandation qu'à la *customization* et l'adaptation de contenu, et nous allons utiliser le terme personnalisation pour indiquer la recommandation du contenu.

Pour une caractérisation automatique du comportement des usagers, les systèmes de personnalisation utilisent généralement l'une des deux approches suivantes [62]. La première approche, appelée filtrage orienté contenu, se base sur le calcul de la similarité existante entre les objets de contenu, souvent appelés *items*, pour la sélection des informations à présenter aux clients (*cf.* section 2.2.1). La deuxième approche, appelée filtrage collaboratif, se base sur l'historique des différents clients et le degré de similarité de leurs préférences (*cf.* section 2.2.2). La combinaison des techniques du filtrage orienté contenu et du filtrage collaboratif donne lieu à une nouvelle catégorie de personnalisation qui est connue sous le nom de filtrage hybride (*cf.* section 2.2.3). Dans la section 2.2.4, nous présentons des travaux récents qui ont porté sur le développement et/ou l'exploitation de mesures de similarité pour les approches de filtrage orienté contenu et le filtrage collaboratif.

Afin de combler certaines lacunes et répondre à des questions soulevées dans ces approches, des techniques du *Web mining* sont utilisées (*cf.* section 2.3). Dans le but d'augmenter la précision et l'efficacité des techniques de personnalisation, l'intégration des techniques du Web Sémantique dans le processus de personnalisation a été étudiée (*cf.* section 2.6). Dans la section 2.7 nous présentons des problèmes qui restent encore posés dans le domaine de la personnalisation Web.

2.2 Approches de la littérature de personnalisation web

Comme indiqué précédemment et selon la source d'information utilisée, on distingue deux approches principales pour la personnalisation Web : le filtrage orienté contenu (*content-based*) et le filtrage collaboratif (*user-based*). Une troisième approche consiste à combiner entre ces deux approches et donne lieu à ce qui est couramment appelé approche hybride.

Afin d'illustrer les différentes approches, nous présentons le tableau 2.1 qui représente un ensemble de séquences d'objets construites à partir d'un ensemble de noms de villes, d'hôtels, de musées et d'endroits à visiter. Une séquence de cet ensemble (séquences d'objets), représente la suite d'objets consultés par un utilisateur lors de sa visite d'un site imaginaire de voyage en ligne. Ainsi, chaque session utilisateur engendre une séquence d'objets. Il est à noter que lorsque l'ordre de parcours des objets n'est pas pris en considération, une session utilisateur sera représentée par un ensemble d'objets au lieu d'une séquence.

Id	Séquence d'objets
u_1	$\langle France, Grenoble, Paris, Louvre, Ritz \rangle$
u_2	$\langle Paris, Louvre, Ritz, InterContinental\_Montreal, Rodin, Mont\_Tremblant \rangle$
u_3	$\langle Australia, Cairns, Clifton\_Beach, Cape\_York\_Safari \rangle$
u_4	$\langle Italy, Roma, Castel\_Sant'Angelo, Grand\_Hotel\_Plaza \rangle$

Tableau 2.1 – Séquences d'objets

2.2.1 Filtrage orienté contenu

La prédiction exploite la similarité existante entre les items. Les usagers se voient affecter un profil qui reflète les items qu'ils ont le plus fréquemment visités. Ainsi, lors d'une visite, ils se voient proposer les items qui sont les plus similaires à ceux de leur profil. La similarité entre items peut être calculée en utilisant différentes mesures qui se distinguent principalement par les connaissances sur les items qui sont prises en considération, et par la façon dont les valeurs associées aux différentes connaissances sont combinées dans les formules (voir sous-section 2.2.4).

Par exemple, en considérant que deux items sont similaires s'ils appartiennent à la même catégorie d'objets, l'utilisateur u_1 du tableau 2.1 qui a déjà visité, entre autres, *Louvre* et *Ritz* se verra recommander par un système de filtrage orienté contenu des hôtels et des musées tels que *The_Westin_Diplomat_Resort_and_Spa*, *Hotel_El_Arz*, *The_National_Gallery*, *Pointe_à_Calière*.

Le filtrage basé sur le contenu est efficace pour produire des recommandations aux utilisateurs avec une grande homogénéité dans leurs préférences. L'autre avantage du filtrage orienté contenu réside dans la facilité d'associer une évaluation numérique (estimation) à un nouvel item qui vient d'être ajouté dans le système. Il suffit généralement de lui affecter l'estimation moyenne des items voisins. En revanche, cette méthode n'est pas efficace avec des utilisateurs ayant des goûts et des préférences variés et sans corrélation. Dans ce cas, les relations (similarités) qui se dégagent des items ne reflèteront pas nécessairement les préférences des usagers. Une autre difficulté des techniques de personnalisation basées sur le contenu réside dans le fait que les informations sur les utilisateurs (ex. âge, sexe, statut social, ...) ne sont pas prises en considération dans le processus de recommandation et de prédiction [85].

2.2.2 Filtrage collaboratif

Le filtrage collaboratif (*FC* ou *Collaborative Filtering*) ou le filtrage social [73] ou encore le filtrage orienté sur l'utilisateur [46] se base sur l'analyse des activités des utilisateurs. Plus concrètement, cela consiste à comparer les différents profils, représentant

le comportement des utilisateurs et à extraire les ressemblances. Ces informations sont ensuite exploitées pour faire des suggestions aux clients et pour la mise à jour de leurs profils. On recommande ainsi à l'utilisateur des items qui ont été visités par d'autres utilisateurs ayant un comportement similaire. À titre d'exemple, il ressort du tableau 2.1 que les utilisateurs u_1 et u_2 ont visité des items en commun (*Louvre* et *Ritz*). Si nous supposons un système de *FC* où deux profils utilisateurs sont considérés similaires s'ils partagent au moins deux items, ce système reconnaîtra u_1 et u_2 comme étant des profils proches et recommandera à l'utilisateur u_1 les autres items qui sont visités par u_2 et qui sont *InterContinental_Montreal*, *Rodin* et *Mont_Tremblant*. Ainsi, le *FC* considère seulement les préférences des utilisateurs sans tenir compte des similarités qui peuvent exister entre les items.

Les différentes approches de filtrage collaboratif sont classées par Breese *et al.* [26] en deux catégories. La première exploite directement les profils des différents utilisateurs dans le processus de recommandation, alors que la deuxième catégorie se base sur un modèle lors du filtrage de l'information.

Filtrage collaboratif orienté mémoire

Le filtrage collaboratif orienté mémoire (*memory-based*) préconise l'utilisation des informations de tous les profils utilisateurs lors du calcul de la similarité entre items [70] (voir sous-section 2.2.4). Dans cette approche, il n'est pas nécessaire de passer par une phase d'apprentissage ou de recherche de motifs pour la construction d'un modèle. L'information existante sur les différents utilisateurs est directement exploitée sous sa forme brute dans le processus de personnalisation. Des mesures statistiques (moyenne, écart type, etc.) sont utilisées pour déterminer les voisins les plus proches de chaque utilisateur actif.

L'avantage de ces modèles réside dans leur évolution dynamique sans nécessiter des traitements supplémentaires liés à la construction/reconstruction d'un modèle. Toutes les nouvelles informations sur un utilisateur sont intégrées de façon naturelle avec les informations existantes. Cependant, le passage à l'échelle est très coûteux en terme de

temps de traitement et d'espace de stockage car toutes les informations collectées sur les utilisateurs sont gardées et utilisées sans aucune compression. On reproche à ce système d'introduire du bruit lors des recommandations à cause du non filtrage des informations qu'on garde dans le profil de chaque utilisateur. Les systèmes de *FC* orientés mémoire font appel à des outils statistiques pour réduire la taille de l'ensemble des profils utilisateurs qui seront pris en considération lors des recommandations. Dans [86], un filtrage des profils permet de réduire le bruit et la redondance des profils. Ainsi, la sélection des profils les plus pertinents est faite en se basant sur les principes suivants : (*i*) sélectionner les profils ayant des préférences différentes les uns des autres et (*ii*) éliminer les profils dont le comportement ne correspond pas aux données auparavant collectées sur le même utilisateur. Dans *PMCF* (*Probabilistic Memory-based Collaborative Filtering*) [85], une approche probabiliste est proposée pour réduire davantage le nombre de profils considérés. Ainsi, les nouveaux utilisateurs se voient remplir un formulaire pour donner leur opinion sur un ensemble d'items. Ensuite, un profil leur est construit et attaché au groupe de profils pour lequel la probabilité d'appartenance est la plus élevée. Les approches orientées mémoires permettent, certes, de garder le maximum d'information sur les comportements des utilisateurs, mais restent peu pratiques en présence de grandes quantités de données. Pour cela, une nouvelle approche de *FC* a vu le jour. Il s'agit du *filtrage collaboratif orienté modèle* où on construit des modèles pour résumer les données.

Filtrage collaboratif orienté modèle

Contrairement à l'approche de *FC* orientée mémoire, l'approche orientée modèle ne garde pas toutes les informations sur les comportements des utilisateurs. Une étape intermédiaire est requise et consiste à construire un modèle des préférences des utilisateurs en s'appuyant sur leurs historiques et à exploiter ce modèle lors de la personnalisation. Le modèle des préférences est généralement construit en se basant sur des techniques d'apprentissage telles que les modèles de graphes associés aux arbres de décisions pour la classification, le groupement, les réseaux de neurones ou les règles d'association. Par

exemple, dans [57] et [68] le modèle représente un réseau bayésien. Généralement, dans ces systèmes, un item est considéré comme un sommet et les arêtes entre deux sommets comme la probabilité de retrouver les deux items dans le même profil. Une nouvelle approche [9] de construction de modèles graphiques considère les utilisateurs comme des sommets et les arêtes pour relier les utilisateurs ayant des profils similaires (voir sous-section 2.2.4). Suite à une phase d'apprentissage durant laquelle le modèle est construit, un arbre de décision est généré pour réaliser des prédictions et des recommandations aux utilisateurs. La performance des systèmes basés sur ces techniques dépend étroitement de la quantité et de la qualité des informations dont on dispose sur le comportement des utilisateurs. En effet, la phase d'apprentissage nécessite la disponibilité de données suffisantes pour obtenir un modèle qui pourra être utilisé efficacement à des fins de personnalisation [70].

Les techniques de groupement se basent sur le partitionnement des utilisateurs en groupes homogènes [78]. Chaque groupe contient alors les utilisateurs ayant des profils similaires. Pour le filtrage de l'information, on présente à un utilisateur les items présents dans les profils de ses voisins dans le même groupe [28]. De même que pour les approches basées sur un modèle, les techniques basées sur le partitionnement nécessitent la disponibilité d'un grand nombre d'utilisateurs et un éventail suffisant de leurs préférences.

L'utilisation des réseaux de neurones pour modéliser les préférences des utilisateurs se résume en deux étapes [38] : le réseau, construit sur plusieurs couches, commence son cycle de vie par une phase d'initialisation et d'apprentissage à partir des données disponibles dans les profils des utilisateurs. Après cette phase d'apprentissage, des couples (utilisateur, item) sont soumis en entrée, et en sortie on aura une valeur de la forme "apprécie", "n'apprécie pas" qui indique si l'item sera proposé (recommandé) à l'utilisateur ou pas.

Un travail de recherche exploitant les règles de dépendance entre les items et les utilisateurs à partir de l'historique des utilisateurs a été proposé dans [38]. Ces règles sont ensuite utilisées pour les prédictions ou les recommandations. Pour un utilisateur donné, on applique les règles déduites de ses voisins (ayant les mêmes préférences que

lui). Ensuite, on lui propose les items présents dans les profils de ses voisins [48].

L'approche *FC* basée modèle a l'avantage de travailler sur des modèles plutôt que sur l'ensemble des données, ce qui peut accélérer le temps de filtrage de l'information lors du processus de recommandation. Cependant, les performances de ces systèmes restent tributaires de la technique choisie pour modéliser le comportement des utilisateurs.

L'avantage de l'approche *FC* est la recommandation d'items qui ont une grande probabilité d'intéresser l'utilisateur. Cependant, ces techniques sont moins efficaces avec les nouveaux items et les nouveaux utilisateurs. Dans le cas de l'ajout d'un nouvel item, ce dernier n'apparaît dans le profil d'aucun utilisateur et ne peut donc être recommandé. De même pour l'ajout d'un nouvel utilisateur, on ne dispose pas de suffisamment d'information dans son profil pour pouvoir retrouver ses voisins [55].

2.2.3 Filtrage hybride

Les systèmes de filtrage orientés contenu s'intéressent aux similarités existantes entre les items. Dans ces approches, les recommandations sont faites en se basant sur les items voisins. Les techniques de *FC*, quant à elles, s'intéressent aux similarités existantes entre les utilisateurs et les recommandations se basent sur les préférences des voisins d'un utilisateur. Afin de bénéficier des avantages de chacune de ces approches, des modèles hybrides ont été développés. Le principe de ces systèmes consiste à utiliser un filtrage orienté contenu qui prend en considération les similarités entre items pour résoudre le problème du nouvel item qui se pose au niveau du *FC*. Un cas d'utilisation, consistera à appliquer le *FC*, pour retrouver un utilisateur ayant les mêmes préférences que l'utilisateur courant, et ensuite à appliquer le filtrage orienté contenu pour retrouver les items similaires à ceux visités par le deuxième utilisateur. Par exemple, à l'utilisateur u_1 du tableau 2.1, un système hybride pourra proposer *Mont_Tremblant* et les deux items *Tala_Guilef* et *Courchevel* qui sont des items similaires (de même catégorie) à *Mont_Tremblant*. Ces items sont visités par l'utilisateur u_2 qui a les mêmes préférences que u_1.

Différentes variantes de systèmes hybrides existants se distinguent par la manière

dont les deux approches (filtrage orienté contenu et *FC*) sont combinées. Par exemple, le système hybride proposé dans [72] consiste à utiliser le *FC* avec une approche probabiliste pour recommander les items déjà existants dans le système, et à utiliser le filtrage orienté contenu pour recommander les nouveaux items. Paulson [58] propose un système où il représente les items et les utilisateurs pas un graphe conceptuel avec le maximum d'attributs possible. Des arcs partent de chaque utilisateur vers les items qu'il a visités. Les similarités entre deux utilisateurs sont calculées en comparant leurs concepts associés en utilisant une mesure de similarité élaborée pour les graphes conceptuels (voir sous-section 2.2.4).

2.2.4 Principales mesures de similarité utilisées

Le calcul de la similarité entre entités est une tâche importante dans le processus de personnalisation [33]. Par exemple, un profil est amené à être comparé avec d'autres profils dans le but d'extraire les ressemblances de comportement existantes. Le calcul de la similarité entre les différents items visités par un même utilisateur peut aussi être une source d'informations pour découvrir les préférences de cet utilisateur. Les valeurs de similarité ainsi obtenues sont directement utilisables pour calculer des prédictions [1] et/ou des recommandations.

Suivant l'approche de personnalisation utilisée, on s'intéresse à un niveau de ressemblance déterminé. Dans le *CF*, ce sont les similarités existantes entre les utilisateurs qui sont les plus pertinentes au processus de personnalisation, alors que dans le filtrage orienté contenu les ressemblances entre objets ont un plus grand intérêt.

Il est cependant à noter que souvent les mêmes mesures de similarité sont utilisées par les systèmes de recommandation qui sont dirigés par le contenu et ceux dirigés par les utilisateurs. Dans les systèmes *FC*, ces mesures sont utilisées pour calculer la similarité entre les vecteurs utilisateurs (chaque entrée du vecteur fait référence à un objet visité) afin de déterminer quel est le vecteur le plus similaire avec le vecteur de l'utilisateur courant. Dans les systèmes de recommandation dirigés par le contenu, les mesures de

1. Prédiction associée à un objet : valeur numérique quantifiant l'intérêt que pourra présenter l'objet pour l'utilisateur.

similarité sont utilisées pour calculer la similarité entre les vecteurs de poids correspondant aux objets du contenu (pour chaque vecteur utilisateur, des poids sont associés aux objets le composant). Les mesures les plus utilisées sont la mesure du *cosinus* [26, 70] et la mesure de corrélation de *Pearson* [67, 73]. La mesure du cosinus calcule la similarité entre deux vecteurs a et b en déterminant l'angle formé par a et b et se calcule par la formule 2.1.

$$SimCosine(a,b) = \cos(\overrightarrow{a}, \overrightarrow{b}) = \frac{\overrightarrow{a} * \overrightarrow{b}}{\| \overrightarrow{a} \|^2 * \| \overrightarrow{b} \|^2} = \frac{\sum_{k=1}^{n} V_{a,k} * V_{b,k}}{\sqrt{\sum_{k=1}^{n}(V_{a,k}^2) * \sum_{k=1}^{n}(V_{b,k}^2)}}, \quad (2.1)$$

où $V_{m,i}$ peut représenter le poids associé à l'item de la position i dans le vecteur de l'utilisateur m. La mesure de corrélation de *Pearson*, quant à elle, est une version centrée de la mesure du *cosinus*, et c'est ainsi qu'elle est aussi appelée *similarité du cosinus ajustée* (*cosine ajusted similarity*). La mesure de corrélation de *Pearson* [26, 29] est utilisée pour quantifier le degré de liaison de deux variables et peut prendre plusieurs formes dont la plus répandue est donnée ci-après.

$$CorrelPearson(a,b) = \frac{\sum_{i=1}^{n}(V_{a,i} - \overline{V_{a,.}}) * (V_{b,i} - \overline{V_{b,.}})}{\sqrt{\sum_{i=1}^{n}(V_{a,i} - \overline{V_{a,.}})^2 * \sum_{j=1}^{n}(V_{b,j} - \overline{V_{b,.}})^2}}, \quad (2.2)$$

où les paramètres peuvent avoir des interprétations différentes suivant l'application. À titre d'exemple, dans le cas du *FC*, $V_{m,i}$ représente l'estimation de l'utilisateur m pour l'item i et $\overline{V}_{m,.}$ correspond à la moyenne de toutes les valeurs estimées par l'utilisateur m pour tous les items.

Une autre façon de calculer la similarité entre les profils de deux utilisateurs ou de deux objets de contenu, consiste à calculer la moyenne du carré de la différence [73] (*mean square difference measure*) des vecteurs des utilisateurs ou des vecteurs des items. Plus la valeur obtenue est petite, plus les deux utilisateurs/items sont similaires (voir formule 2.3).

$$MeanSquare(a,b) = \sqrt{\frac{\sum_{i=1}^{n}(V_{a,i} - V_{b,i})^2}{n}} \quad (2.3)$$

Dans le cas du filtrage orienté contenu, on recherche davantage les similarités existantes entre les différents items qui sont soit visités par un même utilisateur soit visités de la même façon par plusieurs utilisateurs. À titre d'exemple, dans le cas où les objets du contenu sont des documents texte ou des objets décrits par du texte [18], une liste de mots clés est extraite de ce texte afin de représenter l'objet de contenu en question. Pour effectuer cette sélection, le poids de chaque mot clé dans le texte est calculé en utilisant une mesure statistique bien connue dans le domaine de la fouille de texte [69] et de la recherche d'information [16], *TF-IDF* (pour *Term Frequency-Inverse Document Frequency*). Le poids d'un item augmente proportionnellement au nombre d'occurrences de ce terme dans le document ou dans la description de l'objet du contenu. Cependant, ce poids est indexé par la fréquence du terme par rapport à tous les termes utilisés pour représenter tous les objets de contenu de la collection (corpus). Cette indexation permet de diminuer ce poids lorsqu'il n'est pas discriminant pour les objets du contenu.

Supposons que l'on dispose de N descriptions d'objets de contenu (documents) et que $f_{i,j}$ est la fréquence d'apparition d'un mot clé i dans un document j. Ainsi, la fréquence normalisée (*TF*) de i dans le document j est donnée par la fréquence de i divisée par la plus grande fréquence des termes contenus dans la description de j. Cette mesure est donnée par la formule 2.4 où z est un mot clé dans le document j. La normalisation de cette mesure veille à ne pas biaiser les objets de contenu qui ont une longue description par rapport au reste de la collection.

$$TF_{i,j} = \frac{f_{i,j}}{max_z f_{z,j}}, \tag{2.4}$$

Comme nous pouvons le constater, la mesure *TF* ne prend pas en considération l'importance du mot clé par rapport au reste des objets de contenu. Afin de mesurer cette importance, *IDF* est souvent utilisée conjointement avec *TF*. La mesure *IDF* calcule l'importance générale du mot clé sur l'ensemble des objets de contenu d'une collection donnée. Comme le montre la formule 2.5, *IDF* est calculée en prenant le logarithme du résultat de la division du nombre de tous les objets de contenu N par le nombre d'objets

de contenu n_i qui contiennent le mot clé i.

$$IDF_i = log\frac{N}{n_i}, \tag{2.5}$$

Ainsi, le poids du mot clé i dans l'objet de contenu j est donnée par le produit de *TF* avec *IDF* (voir Formule 2.6).

$$w_{i,j} = TF_{i,j} * IDF_i, \tag{2.6}$$

Pour des fins de recommandation, la mesure *TF-IDF* peut être exploitée de la manière suivante :

- Chaque objet de contenu est représenté par un vecteur de p éléments où chaque case correspond au poids d'un mot clé donné dans cet objet.
- À chaque utilisateur est associé un vecteur représentant les poids des mots clés suivant les différentes pages qu'il a visitées. Le calcul le plus simple de ce vecteur consiste à combiner les vecteurs des différents objets de contenu qu'il a déjà visités et de calculer leur moyenne.
- Un objet est suggéré à un utilisateur si le vecteur correspondant à cet objet se rapproche plus du vecteur de l'utilisateur courant comparativement aux autres objets de contenu. Pour calculer la similarité entre deux vecteurs, les mesures du *cosinus* (voir la formule 2.1) et de *Pearson* sont utilisées.

Dans les paragraphes qui suivent, nous présentons les grandes lignes de quelques travaux sur la recommandation qui se basent sur le calcul de la similarité entre profils utilisateurs ou items. Alors qu'une partie de ces travaux se base sur la fréquence de visite des items, d'autres exploitent des descriptions associées à ces items. Nous abordons dans ce paragraphe les techniques les plus récentes.

Manco et *al.* [49] proposent une nouvelle approche de personnalisation qui combine le *FC* et la personnalisation orientée contenu en se basant sur deux mesures de similarité. La première sert à calculer la similarité entre les pages Web, et la deuxième sert au calcul de la similarité entre les sessions. Dans cette approche, chaque page est représentée par un vecteur de poids associé aux items qu'elle contient. La similarité des pages Web

est calculée en utilisant les vecteurs associés ainsi qu'une mesure reflétant la distance de ces pages par rapport au vecteur des pages visitées par un utilisateur. La similarité de deux sessions se base sur les similarités des pages Web consultées au cours de ces sessions. Les sessions sont ensuite regroupées selon leur similarités, et un utilisateur se voit recommander les items appartenant au groupe de sessions proche de son comportement.

Dans [52], on propose un cadre applicatif pour la personnalisation combinant le filtrage collaboratif (*FC*) et le filtrage orienté contenu où un vecteur représentant les évaluations de tous les items par un utilisateur est appliqué. Étant donné l'incomplétude et donc l'éparpillement (*sparsity*) des données, un vecteur n'est que rarement complet. Pour cette raison, les auteurs proposent d'utiliser le filtrage orienté contenu en implantant un classificateur bayésien naïf afin de compléter les vecteurs par des votes virtuels. Suite à l'application de ce "dopage", on obtient des vecteurs complets. Finalement, le *FC* est appliqué pour générer des recommandations en se basant sur les voisins d'un utilisateur.

Dai et *al.* [33] mesurent la similarité de deux sessions en exploitant la ressemblance des attributs des items de chaque session. L'utilisateur, représenté par sa session courante, se voit recommander les items des sessions les plus similaires à la sienne. Une autre mesure est proposée par Mobasher et *al.* [56] pour retrouver la similarité globale de deux items en combinant la similarité des votes de ces items (évaluations explicite de la pertinence des items par les utilisateurs) avec la similarité sémantique de ces items. Pour mesurer la similarité des évaluations, représentant la variance dans les votes des utilisateurs, la formule de *Pearson* présentée par la formule 2.2 est utilisée. La similarité sémantique, quant à elle, est calculée en utilisant la mesure de similarité *cosinus* standard [71] (voir la formule 2.1). Dans cette mesure, les poids des attributs des items sont les seuls éléments utilisés. La même mesure est utilisée par Jin et *al.* dans [45] pour calculer la similarité sémantique de deux items.

Dans cette section, nous avons présenté des mesures de similarité ainsi que des travaux portant sur l'exploitation de ces mesures pour la personnalisation Web. L'utilisation de ces mesures est pratique pour la comparaison des profils d'utilisateurs ou des objets de contenu. Cependant, les mesures utilisées pour la personnalisation Web, ne prennent

pas en considération toutes les connaissances du domaine pouvant servir à représenter le contenu. De telles connaissances peuvent être retrouvées dans les ontologies du domaine. En effet, des objets non identiques peuvent avoir des similarités si les différentes descriptions liées au domaine sont prises en considération. Par conséquent, les mesures de similarité existantes, et qui sont exploitées pour la personnalisation Web, gagneraient à être étendues pour prendre en considération le maximum de connaissances disponibles.

2.3 Prospection du Web : *Web mining*

En incorporant le filtrage orienté contenu au *FC*, on arrive à résoudre le problème du nouvel item, voire du nouvel utilisateur. Cependant, d'autres informations relatives aux transactions de l'utilisateur sur le Web restent non exploitées [8]. En effet, avec les approches classiques, la caractérisation du comportement de l'utilisateur se limite aux items qu'il a déjà visités. D'autres informations du Web méritent d'être regardées de près afin de mieux répondre aux besoins des clients. Il s'agit entre autres de prendre en considération les informations de navigation dans le site (enchaînement chronologique des pages visitées) ainsi que les informations sur les préférences de l'utilisateur du point de vue présentation (présentation des objets des pages Web). Ces deux points sont pris en charge par les techniques du *Web mining*, la discipline qui couvre les méthodes d'extraction de connaissances à partir d'une collection de données en provenance du Web.

Le *Web mining* consiste à adapter les méthodes de fouille de données au contexte du Web. Dans le cadre du *Web mining*, on distingue : (i) la prospection du contenu (*content mining*), (ii) la prospection de la structure (*structure mining*) et (iii) la prospection des usages (*usage mining*) [47]. La prospection du contenu est couverte par le domaine classique de la fouille de données, consistant entre autres à classifier les objets Web, extraire des modèles de représentation des données et des mots clés pour l'indexation et la recherche. La prospection de la structure s'intéresse à la topologie d'interconnexions des objets Web (rangement et classification des pages/sites Web). La prospection des usages, quant à elle, s'occupe de la recherche et de la prédiction de motifs comportementaux, ainsi que des informations de navigation et de parcours des utilisateurs en analysant les

fichiers journaux (*log*) des serveurs ou en suivant en temps réel l'activité de l'utilisateur. Nous nous intéressons particulièrement à cette dernière catégorie du *Web mining* qui permet de caractériser le comportement dynamique d'un utilisateur.

Pour découvrir les motifs de comportement des utilisateurs, la fouille de l'usage du Web fait appel à des méthodes d'apprentissage machine dont les plus utilisées sont [62] :

- la classification : arbres de décision, réseaux bayésiens, réseaux de neurones, théorie des ensembles approximatifs, *etc*,
- le regroupement : partitionnement, création de structure hiérarchique de *clusters*, représentation de groupes par un modèle mathématique,
- la découverte de motifs séquentiels (liste ordonnée d'items qui apparaissent ensemble) avec la prise en considération du facteur temps dans le processus de personnalisation,
- et les règles d'association.

Plusieurs approches ont été proposées pour l'exploitation des techniques de fouilles du Web à des fins de personnalisation. Dans [17], des techniques de la fouille du Web sont utilisées pour l'analyse des fichiers journaux afin d'extraire des motifs de comportement des utilisateurs. Le projet *ClickWorld* [17] cherche à prédire le sexe de l'utilisateur et son intérêt pour une partie du site. Les données récupérées des fichiers journaux sont filtrées, pré-traitées à la recherche de deux catégories d'informations : (i) les informations de navigation (liens d'intérêt) et (ii) les informations sur le contenu des objets que l'utilisateur visite. Par la suite, un modèle d'arbre de décision est construit pour guider les prédictions. Les auteurs de [35] font appel à des techniques de fouille de données pour la construction des profils utilisateurs. Des algorithmes à la *Apriori* [10] sont utilisés pour l'extraction des règles d'association et *CART* (*Classification And Regression Trees*) utilise les arbres de décision pour l'identification de règles de classification. Ces règles permettent de classifier les liens hypertextes des ressources en cinq classes : *recommandé, pas recommandé, deviendra compréhensible, peut être compris, déjà visité.*

Il est à signaler que les règles sont recherchées dans les données de chaque utilisateur. Ce sont donc des règles liées au comportement individuel de l'utilisateur qui sont

communément utilisées pour les recommandations. Le système ne prend pas en considération le comportement de groupes d'utilisateurs (règles extraites à partir de l'historique de navigation de plusieurs utilisateurs). Toutes les règles extraites sont validées directement ou indirectement par un expert.

Zho *et al.* proposent dans [92] une approche de recommandation qui prend en considération le parcours de l'utilisateur sur le site en utilisant une méthode basée sur l'analyse en composantes principales (ACP) qui recherche les principaux facteurs qui caractérisent les relations entre les pages visitées. Ces facteurs sont ensuite utilisés pour créer des motifs de comportement de navigation. Dans un tel système, les sessions représentent les observations et les variables modélisent les pages visitées. L'ACP est donc utilisée pour extraire les relations qui caractérisent des groupes de pages.

Dans [83], les auteurs présentent une approche de recommandation basée sur les motifs séquentiels dont les plus fréquents sont stockés sous forme d'une arborescence. La recommandation s'effectue en cherchant la correspondance d'une sous-séquence de l'utilisateur courant avec une branche de l'arbre. Les items à recommander sont ceux présents dans le reste de la branche. Cette approche a l'avantage d'éviter de sauvegarder toutes les sessions utilisateurs et ainsi réduire le temps de comparaison entre la session courante et le reste des sessions.

Dans le cadre de ce travail, nous nous intéressons à l'extension d'une catégorie bien spécifique des techniques de fouille de l'usage du Web et qui est la fouille des motifs séquentiels à partir de données décrites à l'aide d'une ontologie du domaine. Le choix des motifs séquentiels pour la représentation du comportement des utilisateurs est justifié par leur nature qui permet de préserver l'information sur l'ordre chronologique de parcours des objets de contenu. Cette information nous permet de connaître l'ordre dans lequel l'utilisateur préfère naviguer dans le contenu, et est capitale dans des applications de recommandation tel que le *e-learning*, où l'ordre de l'accès aux ressources peut par exemple refléter le degré de difficulté dans l'apprentissage de la ressource en question.

2.4 Évolution de la recherche sur les motifs séquentiels

Le problème de recherche de motifs séquentiels a été introduit par Agrawal et Srikant [11]. Les motifs séquentiels, tels que proposés dans leur premier travail, sont composés d'items seulement. La définition initiale a été étendue pour inclure les catégories du domaine (les classes), et le résultat est l'apparition d'une nouvelle catégorie de motifs : *les motifs séquentiels généralisés* [76]. Ces motifs intègrent plus de connaissances du domaine que leurs prédécesseurs. L'intégration des taxonomies d'objets dans le processus de fouille permet de détecter des motifs à différents niveaux d'abstraction pour résumer un ensemble de données. Prendre en considération les taxonomies d'objets permet de détecter des motifs qui ne sont pas visibles au niveau objet. En effet, des sous-séquences d'objets non fréquentes peuvent devenir fréquentes si les objets sont remplacés par les catégories auxquelles ils appartiennent. Par exemple, si nous considérons les séquences $\langle Paris, Ritz \rangle$ et $\langle Roma, Grand\_Hotel\_Plaza \rangle$, les objets des deux séquences sont à priori différents et aucune ressemblance ne peut être tirée si on raisonne au niveau objet. Cependant, en remplaçant les objets par leur classes (catégories) d'appartenance, dans les deux cas on obtient la même séquence, à savoir $\langle Capital, Luxury\_Hotel \rangle$. On constate qu'au niveau objet, il y a un manque de connaissances qui voile des ressemblances existantes à d'autres niveaux d'abstraction.

Mannila et *al.* [50] ont proposé une (autre) approche où les motifs sont composés d'épisodes. Un épisode est défini comme une collection d'événements qui se reproduisent proches les uns des autres dans une période de temps relativement courte et suivant un certain ordre partiel. Au lieu de chercher les motifs simples, Pinto et *al.* [63] ont proposé la recherche de motifs séquentiels multidimensionnels (*multidimentional sequential patterns*) où les motifs sont extraits de données multidimensionnelles (chaque dimension représente les valeurs d'un attribut). Le principe de cette approche (fouille de motifs séquentiels multidimensionnels) consiste à caractériser chaque motif par les attributs dont les valeurs dépendent du contexte des transactions (informations sur les utilisateurs, période de l'année, heure, etc.) afin de classifier ces motifs. Bettini *et al.* [24] ont proposé des approches pour la recherche des règles d'association à partir de

motifs séquentiels où la prémisse et la conclusion sont chronologiquement ordonnés. Wang *et al.* [81, 84] ont proposé et étudié le problème de méta-motifs sur les données séquentielles, où un méta-motif est un motif de motifs.

Récemment, des cadres de fouille qui prennent en considération un ou plusieurs éléments du problème de fouille de motifs fréquents ont été proposés. Zaki *et al* ont proposé dans [91] un cadre de fouille pour la recherche de motifs fréquents qui inclut les itemsets[2], les séquences et les graphes. Ce cadre se limite à couvrir les approches où seulement les motifs d'objets sans généralisation sont recherchés. En effet, il ne tient pas compte des représentations riches des connaissances du domaine, tel que les relations, et le critère d'intérêt se limite à la fréquence minimale d'apparition des items, appelé aussi *support minimal.*

Berendt a proposé un cadre de fouille pour retrouver les motifs fréquents qui prend en considération la généralisation entre les noeuds de graphes [22]. Dans [42], Horvath *et al.* proposent un cadre de fouille d'hypergraphes fréquents.

L'utilisation des techniques du *Web mining*, comme on vient de le voir, permet d'extraire les motifs de comportement des utilisateurs et de réduire l'espace des données considérées. Cependant, les techniques de *Web mining* actuelles ne permettent pas de capturer les relations complexes entre objets et d'analyser avec un haut niveau de granularité les liens et les ressemblances qui peuvent exister entre les objets inter-connectés. En effet, les approches existantes sont généralement limitées à la recherche d'objets ou de classes alors que d'autres connaissances, telles que les liens inter-objets et les relations inter-classes, ne sont pas prises en considération. De telles connaissances peuvent s'avérer intéressantes entre autres pour la compréhension du passage d'un objet à un autre dans la séquence et apporter plus de précision aux séquences de classes qui non seulement gagnent en abstraction mais aussi en précision. En effet, ajouter des relations sémantiques entre les classes d'une séquence fera réduire le nombre de séquences d'objets qui sont instances de cette séquence de classes aux instances les plus pertinentes seulement.

Tenir compte des liens et des relations dans le processus de recherche conduit à

2. Itemset : ensemble d'items (d'objets).

la nécessité de manipuler des structures complexes apparentées à des graphes orientés. Ainsi, de nouvelles techniques de recherche doivent être développées, comme c'est le cas pour la recherche de motifs dans des graphes, avec la prise en considération des connaissances du domaine en même temps que de l'ordre chronologique des éléments d'une séquence. Une solution possible pour combler ces lacunes consiste à exploiter les techniques développées autour du Web sémantique qui offrent une représentation riche du contenu (*cf.* section 2.6) dont l'intégration dans le processus de personnalisation devrait permettre d'améliorer les recommandations.

2.5 *Apriori* pour la fouille de règles d'association et de motifs séquentiels

Une approche naïve de recherche de règles d'association, à partir d'une base d'itemsets [3], consistera à rechercher tous les itemsets dont la fréquence d'apparition est supérieure ou égale au support minimal. Par la suite, les règles d'association intéressantes seront générées à partir des itemsets fréquents. Une règle est dite intéressante si sa valeur de confiance [4] est supérieure ou égale à un seuil minimal.
Cette approche génère tous les itemsets possibles. Ainsi, pour n items, $\sum_{k=1}^{n} \frac{n!}{(n-k)!}$ itemsets différents seront générés. À titre d'exemple, si $n = 40$ alors l'approche aura à générer 1631830566495795468691222539194664690552176640756 itemsets.

Afin de ne pas générer tous les itemsets, *Agrawal* et *al.* ont proposé l'algorithme *Apriori* [10] qui évite de parcourir tout l'espace des itemsets.
Pout éviter de générer tous les itemsets possibles, *Apriori* se base sur la propriété d'antimonotonie de la fréquence pour élaguer l'espace de recherche d'itemsets. Cette propriété s'énonce comme suit : si un itemset A n'est pas fréquent, alors tous les itemsets qui sont plus spécifiques que A ne sont pas fréquents. Par exemple, si l'itemset $A = \{Grenoble, Paris\}$ n'est pas fréquent, alors l'itemset $B = \{Grenoble, Paris, Louvre\}$

3. Un itemset est un ensemble d'items.
4. La confiance d'une règle $X \rightarrow Y$ est la proportion des enregistrements contenant la conclusion Y parmi ceux qui contiennent la prémisse X.

n'est pas fréquent.

L'idée clé de l'algorithme *Apriori* et de générer les itemsets fréquents par niveau où chaque niveau est composé des itemsets de même cardinalité (la cardinalité d'un itemset est le nombre de ses items). Ainsi, lors de la phase d'initialisation, l'algorithme génère l'ensemble des itemsets fréquents de cardinalité 1 et par la suite calcule récursivement les itemsets fréquents dont la cardinalité varie de 2 jusqu'à n.
Les itemsets fréquents d'un niveau k ($k \in [2..n]$) sont générés à partir des itemsets fréquents de niveau $k-1$ en deux phases. Durant la première phase, tous les itemsets de niveau k sont générés en combinant deux à deux les itemsets de niveau $k+1$ qui ne différent que d'un seul item. À titre d'exemple, si nous considérons que les deux itemsets $\{Grenoble, Paris\}$, $\{Paris, Louvre\}$ sont fréquents, leur combinaison donnera lieu à l'itemset $\{Grenoble, Paris, Louvre\}$. Ces itemsets sont appelés itemsets candidats. Dans la deuxième phase, un test de fréquence est effectué sur chaque itemset candidat et seulement ceux qui sont suffisamment fréquents sont gardés (voir le pseudo code de l'algorithme 1). L'algorithme se termine lorsqu'une itération ne produit aucune itemset fréquent ou que aucun itemset candidat ne peut être généré de l'ensemble des itemsets fréquents du niveau infereieur.

Une fois que tous les itemsets fréquents sont générés, des règles d'association en sont déduites. Le processus de génération de ces dernières est comme suit. Pour chaque itemsets fréquent X, chaque sous-ensemble est choisi comme prémisse d'une règle et le reste des items de X deviennent la partie conclusion. Comme X est fréquent, alors tous les sous-ensembles de X le sont. Connaissant les fréquences des différents sous-ensembles, il est possible de calculer la confiance d'une règles et de ne la garder que si sa confiance est supérieure ou égale à un seuil minimal.

Pour illuster le processus de fouille de règles d'association, nous considérons la base de quatre itemsets composés d'objets de l'ontologie *Travel* (voir le tableau 2.2). Ces itemsets sont les données d'entrée du processus de fouille des itemsets fréquents. Le support minimal considéré est 2.

Lors de la phase d'initialisation, la fréquence de tous les items de la base est calculée (tableau 2.3) et seulement ceux qui sont suffisamment fréquents sont gardés. Le résultat

Algorithm 1 Apriori

1: **Entrée :**
2: I; ▷ Ensemble d'items
3: $\mathscr{D}$, σ ; ▷ Ensemble de transaction d'items et seuil minimal du support

4: **Sortie :**
5: $\mathscr{F}_\sigma$; ▷ Ensemble d'itemset fréquents

6: **Initialisation :**
7: $\mathscr{F}_\sigma^1 \leftarrow$ { items de I qui sont fréquents} ;

8: **Méthode :**
9: **for** $(k = 2; \mathscr{F}_\sigma^{k-1} \neq \emptyset; k++)$ **do**
10: $\mathscr{F}_c^k \leftarrow$ itemsets candidats générés à partir de $\mathscr{F}_\sigma^{k-1}$;
11: calculer le support de chaque élément de $\mathscr{F}_c^k$;
12: $\mathscr{F}_\sigma^k \leftarrow$ éléments de $\mathscr{F}_c^k$ dont le support est supérieur ou égal à σ ;
13: **end for**
14: **return** $\bigcup_{i=1,k-1} \mathscr{F}_\sigma^i$;

Id	Itemset
d_1	$\{Paris, Louvre, Grenoble\}$
d_2	$\{Ritz, Louvre, Rodin\}$
d_3	$\{Paris, Ritz, Louvre, Rodin\}$
d_4	$\{Ritz, Rodin\}$

Tableau 2.2 – Ensembles d'objets (itemsets)

de cette phase est représenté par le tableau 2.4.

Dans une première itération, les itemsets candidats de taille 2 sont générés à partir des itemsets fréquents de taille 1 du tableau 2.4 (tableau 2.5). Par la suite, seulement les itemsets candidats ayant une fréquence supérieure ou égale à deux sont gardés, et ces derniers constituent les itemsets fréquents de taille 2 (tableau 2.6).

Lors de la deuxième itération, les itemsets candidats de taille 3 sont générés à partir des itemsets fréquents de taille 2 du tableau 2.6 (tableau 2.7). L'ensemble des itemsets fréquents de taille 3 du tableau 2.8 est composé des itemsets candidats de taille 3 du tableau 2.7.

À la troisième itération, l'algorithme s'arrête car aucun itemset candidat de taille 4 ne peut être généré de l'unique itemset fréquent $\{Ritz, Louvre, Rodin\}$.

Les règles d'associations peuvent maintenant être générées à partir des itemsets fréquents qui contiennent plus d'un item. Par exemple, l'itemset $\{Paris, Louvre\}$ du tableau 2.6 nous permet de générer les deux règles d'association $r_1 = Paris \rightarrow Louvre$, $r_2 = Louvre \rightarrow Paris$. La confiance de la règle r_1 est 100% car tous les itemsets de départ (du tableau 2.2) qui contiennent la prémisse *Paris* contiennent aussi la conclusion *Louvre* (3/3). Par contre, la confiance de la règle r_2 n'est que de 66%. En effet, sur les trois itemsets de départ qui contiennent la prémisse *Louvre* seulement deux comportent *Paris* (2/3).

Agrawal et *Srikant* on adapté l'algorithme *Apriori* pour la fouille de motifs séquentiels fréquents [11]. La principale modification de l'algorithme de base consiste à remplacer les itemsets par des listes (ou séquences) d'items.

Itemset	Support
$\{Paris\}$	2
$\{Ritz\}$	3
$\{Louvre\}$	3
$\{Grenoble\}$	1
$\{Rodin\}$	3

Tableau 2.3 – Itemsets candidats de taille 1

Itemset	Support
$\{Paris\}$	2
$\{Ritz\}$	3
$\{Louvre\}$	3
$\{Rodin\}$	3

Tableau 2.4 – Itemsets fréquents de taille 1

Itemset	Support
$\{Paris, Ritz\}$	1
$\{Paris, Louvre\}$	2
$\{Paris, Rodin\}$	1
$\{Ritz, Louvre\}$	2
$\{Ritz, Rodin\}$	3
$\{Louvre, Rodin\}$	2

Tableau 2.5 – Itemsets candidats de taille 2

Itemset	Support
$\{Paris, Louvre\}$	2
$\{Ritz, Louvre\}$	2
$\{Ritz, Rodin\}$	3
$\{Louvre, Rodin\}$	2

Tableau 2.6 – Itemsets fréquents de taille 2

Itemset	Support
$\{Ritz, Louvre, Rodin\}$	2

Tableau 2.7 – Itemset candidat de taille 3

Itemset	Support
$\{Ritz, Louvre, Rodin\}$	2

Tableau 2.8 – Itemset fréquent de taille 3

2.6 Web sémantique et personnalisation

Le Web sémantique est une discipline récente qui vise à faciliter l'exploitation et l'interprétation de l'information par les machines. Plus précisément, l'un de ses objectifs est d'offrir aux machines la possibilité de partager et d'interpréter les grandes masses d'informations disponibles sur le Web sans l'intervention de l'être humain. Les ontologies sont un des outils de représentation des connaissances dans le cadre du Web sémantique. Dans une ontologie, on a des concepts, une structuration d'objets complexes et les relations existantes entre ces objets. L'intégration des outils du Web sémantique dans la personnalisation devrait permettre l'explication, l'interprétation et le raisonnement automatique sur les données utilisateur qui peuvent être hétérogènes. Les ontologies permettent de caractériser et de donner une interprétation aux règles (motifs) de comportement découvertes par le *Web usage mining*. Par exemple, un visiteur qui a montré un intérêt pour les tableaux de *Picasso* se voit classiquement recommander seulement les tableaux de ce peintre, alors qu'une ontologie du domaine peut entraîner la recommandation non seulement des tableaux de *Picasso* mais aussi ceux d'autres peintres qui traitent des mêmes thèmes ou qui sont de la même génération ou du même style que *Picasso*. La prise en compte de l'aspect sémantique des objets offre un niveau de granularité très élevé dans la caractérisation des objets recommandés.

Dai *et al.* abordent dans [33] les différentes perspectives de l'intégration des connaissances sémantiques dans les différents niveaux du processus de personnalisation (prétraitement, extraction de motifs, post-traitement). À chacun de ces niveaux, les connaissances du domaine peuvent être intégrées de la façon suivante : (i) dans la phase de prétraitement, on fait la correspondance entre les données transactionnelles d'un côté et les concepts et objets de l'ontologie du domaine de l'autre côté, (ii) durant l'extraction de motifs, on crée des représentations agrégées des entités ou des groupes d'entités qui sont accédés par les utilisateurs pour modéliser leur comportement, et (iii) dans la dernière phase qui est le post-traitement, les motifs découverts sont filtrés et interprétés en se basant sur l'ontologie du domaine. Dans la même optique, Dai et Mobasher ont proposé dans [32] une nouvelle technique de filtrage orienté contenu pour produire

des recommandations en se basant sur les ontologies. L'idée de base consiste à calculer la similarité entre les items sélectionnés dans une session utilisateur au niveau attribut. Les items (objets) et leurs attributs sont représentés par une ontologie du domaine. L'approche de découverte de connaissances comportementales proposée [32] apporte certes des améliorations pour affecter une évaluation aux nouveaux items mais se base sur des recommandations individuelles qui laissent le problème du nouvel utilisateur toujours posé.

D'autres travaux, portant sur l'intégration des techniques du Web sémantique dans la personnalisation, proposent d'utiliser les ontologies différemment. Ainsi, dans [35] et [66] les métadonnées sont classées selon leur catégorie : informations sur les documents, métadonnées du domaine, modèle utilisateur, etc. Dans ces techniques, chaque catégorie est représentée par une ontologie.

Dolog *et al.* [35] proposent un système où le domaine, les documents, les utilisateurs et les faits (sessions des utilisateurs) sont modélisés par des ontologies sur lesquelles un raisonnement est possible. À des fins d'inférence, d'interrogation et de transformation sur ces ontologies, un langage à base de règles, appelé *TRIPLE*, est utilisé. Toujours en exploitant les outils du Web sémantique, Razmerita *et al.* [66] proposent une architecture de modélisation des utilisateurs basée sur les ontologies pour les systèmes de gestion de connaissances (*KMS*).

Récemment, Berendt [20, 21] a proposé d'intégrer les taxonomies (hiérarchies) de concepts dans le processus de recherche de sous-structures fréquentes à partir d'une base de graphes correspondant à des transactions Web. Ainsi, l'approche proposée recherche des motifs de graphes avec des noeuds pouvant avoir des noms de concepts appartenant à différents niveaux d'abstraction d'une taxonomie. Cependant, l'approche proposée se limite à la prise en considération de la généralisation entre concepts de l'ontologie seulement, alors que la généralisation entre les relations de l'ontologie n'a pas été prise en charge.

D'un autre côté, les auteurs de l'article [36] proposent d'intégrer l'ordre entre entités de l'ontologie en tant qu'extension au langage de description d'ontologies OWL-DL. Cependant, un langage OWL-DL qui prend en charge l'ordre de séquencement des entités

n'élimine pas le besoin d'un langage spécifique pour la description de motifs complexes si ce n'est la possibilité offerte pour représenter avec plus de flexibilité un domaine d'application donné.

2.7 Questions ouvertes

Malgré les nombreux travaux effectués dans le domaine de la personnalisation Web, il reste des questions ouvertes. Voici quelques points soulevés par Pierrakos *et al.* [62] et Adomavicius et *al.*.

- Mise à l'échelle : la plupart des techniques utilisées pour modéliser ou agréger le comportement de l'utilisateur sont soit coûteuses en temps de calcul et en mémoire, soit difficiles à appliquer sur de grandes quantités de données.
- Mise à jour incrémentale : le comportement des utilisateurs est dynamique et est sujet aux changements. Il en est de même du contenu des sites où de nouvelles données peuvent être ajoutées ou supprimées dynamiquement. Le défi est d'intégrer de façon incrémentale ces données dans un modèle existant du système de personnalisation. Des algorithmes existants prennent partiellement en charge cette problématique mais l'ordre d'apparition des données influence ces algorithmes.
- Prise en charge du temps dans le processus de découverte de données : ceci suggère par exemple la prise en considération de l'ordre chronologique des transactions lors de la recherche de modèles de comportement des utilisateurs, ou la recherche de co-occurrences au sein d'une fenêtre temporelle.
- Non intrusivité : les informations à récolter sur les utilisateurs doivent en premier lieu ne pas porter atteinte à leur vie privée et devraient être récoltées de manière transparente à l'utilisateur sans complexifier la tâche de navigation par une sollicitation inadéquate ou abusive.

Une autre limite, plus conceptuelle que technique, réside dans la manière dont la prédiction du comportement futur d'un utilisateur est faite. Dans [6, 7], les règles de comportement d'un utilisateur sont déduites uniquement de ses préférences (profils statiques) et de ses transactions passées. Avec cette approche, trois problèmes se posent :

(i) le problème du nouvel utilisateur, (ii) l'impossibilité de recommander des items qui ne sont pas présents dans le profil utilisateur mais qui peuvent présenter un intérêt pour lui, (iii) l'utilisateur se voit toujours proposer les items d'un groupe restreint limitant ainsi son champ de visibilité du système Web. Ces trois points sont comblés en prenant en considération les profils des autres utilisateurs lors de la recherche des motifs de comportement qui vont servir à effectuer des recommandations. La dernière question soulevée concerne l'intégration des connaissances sémantiques dans le processus de personnalisation. Quelques travaux récents ont abordé cette question, par exemple, Dai *et al.* [33] proposent d'intégrer les connaissances sémantiques d'un domaine dans les différentes étapes de la personnalisation, allant du prétraitement au post-traitement en passant par la phase de découverte de motifs. Dans le même article, ils signalent la nécessité de développer des techniques de prospection de données qui prennent en considération les connaissances du domaine. De notre côté, nous posons la question suivante : est-ce-que la prise en considération des connaissances d'un domaine dans les approches existantes sera suffisante ou faudra-t-il développer de nouvelles approches construites autour des connaissances du domaine ?

D'un autre côté, les méthodes de calcul de similarité actuelles utilisées dans le cadre de la personnalisation Web présentent des limites. Souvent, seuls les attributs sont utilisés pour mesurer la similarité entre deux items, alors que les liens entre les items et leurs classes d'appartenance ne sont pas exploités dans ces mesures. Une telle vision réduit la flexibilité du système de personnalisation et restreint son application aux sites avec un contenu de même type et homogène. Cependant, deux objets non identiques peuvent être similaires en se basant sur leurs classes et sur leurs liens tels que modélisés dans une ontologie du domaine. Mettre au point une méthode de calcul de similarité qui prend en considération la structuration des items et leurs classes dans l'ontologie du domaine est une nouvelle orientation qui permettra de découvrir des ressemblances avec des niveaux de granularité plus élevé.

Dans le cadre de cette thèse, nous nous intéressons à la prise en considération des connaissances du domaine dans la personnalisation Web. Plus précisément, nous proposons une approche qui intègre les connaissances disponibles sur les objets du contenu

dans le processus de personnalisation.

2.8 Conclusion

Dans ce chapitre, nous avons présenté les principales approches utilisées dans le domaine de la personnalisation Web ainsi que l'évolution de la discipline. La figure 2.1 représente une vue globale des approches qui sont utilisées, ou pourront être utilisées, pour la personnalisation Web. Les flèches indiquent le sens des appels d'utilisation d'une catégorie d'approches à partir d'une autre catégorie d'approches. Par exemple, les approches de filtrage de contenu peuvent utiliser des mesures de similarité ou des techniques de fouille du Web afin d'effectuer des recommandations. Le triangle du milieu représente la représentation des données manipulées en utilisant les connaissances du domaine disponibles.

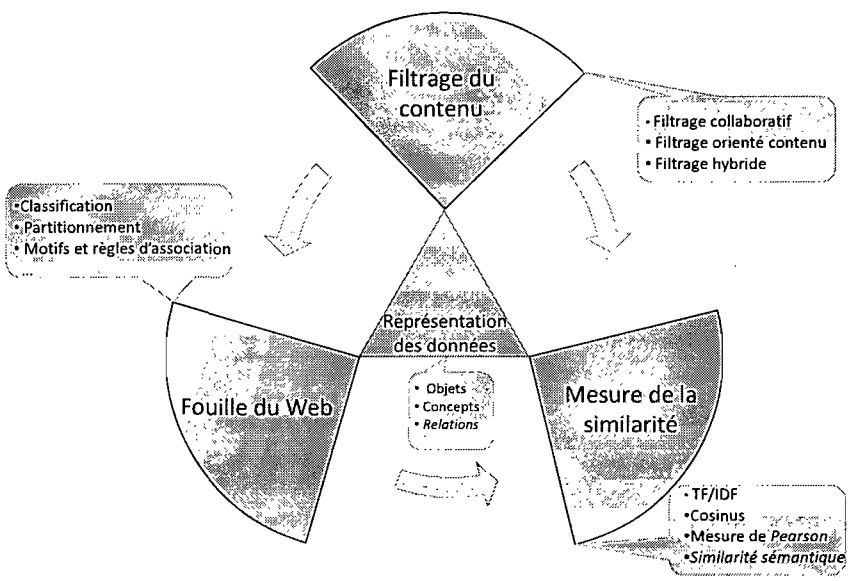


Figure 2.1 – Approches utilisées pour la personnalisation Web.

Chaque approche a ses avantages et ses inconvénients. Combiner les différentes techniques existantes semble mieux traiter le problème. À cela s'ajoute le pouvoir de des-

cription des liens entre objets ainsi que la structuration des objets complexes qui peuvent procurer à l'intégration des connaissances du domaine une place prépondérante dans le processus de personnalisation. Une connaissance riche et complète d'un domaine ne peut que contribuer à l'amélioration du rendu de la personnalisation. Ceci peut passer par l'intégration de ces connaissances dans le processus de caractérisation du comportement des utilisateurs. Un comportement qui peut être représenté par des motifs extraits des données d'usage. Notre travail consiste alors à définir une catégorie de motifs séquentiels où des représentations riches des connaissances sont intégrées.

Chapitre 3

Représentation des connaissances avec les ontologies

Afin de pouvoir exploiter une connaissance, cette dernière devrait au préalable être représentée sous une forme donnée. La représentation des connaissances consiste à exprimer et structurer les connaissances d'un domaine à l'aide d'un ensemble de formalismes.

Différentes formes et divers formalismes pour la représentation des connaissances ont été élaborés. Dans un premier temps, des représentations de connaissances s'inspirant de l'organisation du cerveau humain (sous forme de graphes) ont été proposées. Par la suite, des approches basées sur des représentations graphiques existantes et des formalismes logiques ont été introduites [15]. Dans les approches, qui à l'origine étaient purement graphiques, on retrouve entre autres, les *frames*, les *réseaux sémantiques* et les graphes conceptuels. Une catégorie complètement fondée sur les bases d'un formalisme logique est celle des logiques de description. En se basant sur un formalisme logique rigoureux, il est plus aisé de raisonner sur des connaissances comparativement aux approches qui sont uniquement graphiques.

Réseaux sémantiques

Les réseaux sémantiques sont un formalisme graphique de représentation des connaissances sous forme de noeuds liés par des arcs. Les noeuds représentent les objets ou les concepts à représenter, et les arcs représentent les propriétés, les relations ou les actions possibles sur ces noeuds [64]. Les arcs sont orientés et sont en général étiquetés. Ainsi, un réseau sémantique est un graphe orienté. Les propriétés sont associées à un noeud par les arcs de propriétés et sont de type simple. Cela signifie qu'une propriété ne peut être une composition d'autres éléments du réseau. Il est à noter que les objets et

les concepts sont distingués dans un réseau sémantique pour éviter toute confusion. Les concepts sont une représentation abstraite des catégories d'objets. La figure 3.1 représente un exemple de réseau sémantique composé de deux concepts (*Museum* et *City*), de deux objets (*Paris* et *Louvre*), ainsi que d'une relation entre les deux objets et la relation d'appartenance d'un objet à un concept donné ($is-instance-of$).

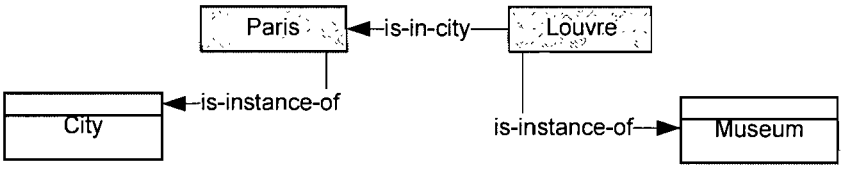


Figure 3.1 – Un exemple de réseau sémantique.

Une revue détaillée des réseaux sémantiques et des différentes variantes de formalismes existant est présentée dans [25].

Frames

Les schémas ou cadres (ou *frames*) sont des structures de données utilisées pour décrire des connaissances d'un domaine donné. L'objectif des frames est de regrouper toute la connaissance sur une situation donnée dans un même "objet" [15]. Les frames et les réseaux sémantiques sont considérés appartenir à la même famille de formalismes, avec la différence que dans les frames on peut définir des propriétés composées (à partir d'autres frames), ce qui n'est pas possible avec les réseaux sémantiques standards. Un frame se définit comme étant une structure de données qui regroupe l'ensemble des connaissances relatives à un concept. Un frame est composé d'un ensemble d'attributs, appelés *slots*, qui correspondent aux propriétés d'un concept. Les attributs d'un frame sont décris par ce qu'on appelle les *facettes*. On distingue deux types de facettes : les facettes déclaratives qui associent des valeurs aux attributs et les facettes procédurales qui décrivent les actions qui sont activées lorsqu'on accède aux valeurs des attributs [54]. Les frames sont organisés en hiérarchies (frames, sous-frames) par héritage d'attributs/facettes. L'exemple ci-dessous illustre l'utilisation de frames pour représenter les

deux concepts *Museum* et *City*, leurs instances respectives *Louvre* et *Paris* ainsi que la relation qui lie ces instances ($is-in-city$).

$Louvre : Frame, \{Slot : is-instance-of \{Museum : Frame\},$
$Slot : is-in-city \{Paris : Frame, Slot : is-instance-of \{City : Frame\}\}\}$

Graphes conceptuels

Les graphes conceptuels constituent un formalisme logique qui permet la description des données sémantiques et le raisonnement pour en déduire de nouvelles valeurs dans des cas de composition. Un des apports du modèle des graphes conceptuels est la séparation entre la connaissance abstraite (concepts, relations, *etc.*) et la description d'une situation [74]. Le but des graphes conceptuels est de créer une représentation graphique des connaissances lisibles par les humains, tout en se basant sur un formalisme utile pour les traitements automatisés, c'est-à-dire pouvoir raisonner sur les connaissances. Ainsi, dans les graphes conceptuels on ne s'intéresse pas seulement à la création de figures graphiques pour représenter une connaissance, mais aussi pour pouvoir raisonner sur cette connaissance.

La figure 3.1 représente un exemple de graphe conceptuel composé de deux noeuds représentant l'objet *Louvre* et son concept d'appartenance (*Museum*), ainsi que l'objet *Paris* et le concept auquel il appartient (*City*). Un troisième noeud qui représente la relation $is-in-city$ qui lie les deux objets *Louvre* et *Paris*. Cet exemple de graphe conceptuel peut être représenté en utilisant la *forme d'échange des graphes conceptuels* comme suit :

$$[Museum : Louvre * x][City : Paris * y](is-in-city?x?y)$$

Le graphe de la figure 3.2 est composé de deux noeuds représentant l'objet *Louvre* et son concept d'appartenance (*Museum*), ainsi que l'objet *Paris* et le concept auquel il appartient (*City*). Un troisième noeud qui représente la relation $is-in-city$ qui lie les deux objets *Louvre* et *Paris*.

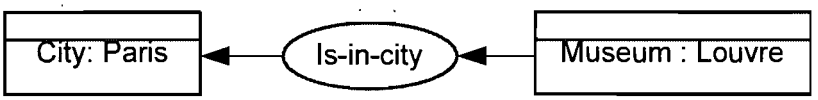


Figure 3.2 – Un exemple de graphe conceptuel.

Logiques de description

Une logique de description est un langage de représentation des connaissances d'un domaine par des entités qui ont une description syntaxique à laquelle est associée une "sémantique". Dans les logiques de description, un concept représente un ensemble d'individus (objets), et un rôle représente une relation binaire entre les individus. Les concepts et rôles (propriétés) ont une structure simple ou élaborée à partir d'un ensemble de constructeurs. Une sémantique est associée à chaque description de concept et de rôle par l'intermédiaire d'une interprétation. La section 3.1 présente avec plus de détails le formalisme des logiques de description. Dans les logiques de description, le fait que le musée *Louvre* se trouve dans la ville de *Paris* sera représenté comme suit. En premier lieu, deux concepts représentant les villes et les musées respectivement sont définis. Ces concepts sont *Museum* et *City*. Pour représentation le fait qu'un musée se trouve dans une ville donnée, le rôle $is - in - city$ est défini comme étant la relation qui lie les objets de type *Museum* avec les objets de types *City*. Pour finir, les objets *Louvre* et *Paris* sont définis comme étant des instances respectives des concepts *Museum* et *City*, et le couple $(Louvre, Paris)$ est considéré satisfaire la relation binaire $is - in - city$ (pour plus de détails voir la section 3.1).

Ces dernières années, un intérêt particulier a été accordé aux logiques de description en tant que formalisme pour la représentation des connaissances d'un domaine particulier [15]. De ce formalisme est né OWL (*Ontology Web Language*), un langage de représentation de connaissances qui offre une sémantique riche du fait qu'il dérive de DAML+OIL [31]. OWL est conçu comme une extension de *Resource Description Framework* (RDF) et *RDF Schema* (RDFS). Une ontologie est le composant principal de

représentation et de gestion de la connaissance dans le cadre du Web sémantique [27].

Dans ce qui suit, nous présentons les notions de base des logiques de description ainsi qu'une introduction aux ontologies et un survol rapide du langage OWL.

3.1 Bases des logiques de description

Les logiques de description (*DL* pour *Description Logics*) sont une famille de languages de représentation de connaissance. Ils sont utilisés pour représenter les connaissances d'un domaine de manière formelle et structurée [15].

Dans les logiques de description, les connaissances sont organisées au moyen d'une base de connaissances. Cette base est composée de deux parties, une première partie pour représenter les concepts génériques et la deuxième partie pour représenter les individus.

La plupart des logiques de description représentent les connaissances en deux parties distinctes. La première partie regroupe les connaissances terminologiques et la deuxième partie concerne les assertions sur les éléments de la terminologie.

Une logique de description est principalement caractérisée par un ensemble de constructeurs qui permettent d'obtenir des concepts et des rôles complexes à partir de concepts et de rôles atomiques. Un concept atomique est une relation unaire qui représente un ensemble d'objets, appelés individus, alors qu'un rôle atomique correspond à une relation binaire entre deux concepts. Un concept (respectivement rôle) est dit primitif s'il n'est pas défini à partir d'autres concepts (respectivement d'autres rôles). Les rôles et concepts ainsi regroupés forment le vocabulaire de la terminologie, appelé aussi domaine d'application, et qui est désigné par le terme *TBox*. La deuxième composante d'une base de connaissance dans *DL*, regroupe l'ensemble des formules relatives aux faits, et est désignée par le terme *ABox*. La composante *ABox* contient des assertions sur les individus par rapport aux concepts et rôles définis dans la *TBox*. Par exemple, pour le domaine du tourisme, *Hotel* et *Destination* sont des concepts qui regroupent respectivement les objets de type hôtel et de type destination de voyage. Un rôle typique dans cet exemple est la relation *hasAccommodation* entre les concepts *Destination* et *Hotel*. L'expression "*Paris est instance de Destination*" est une assertion qui indique que l'individu *Paris* est

instance du concept *Destination*.

La sémantique formelle des concepts et des rôles dans un langage *DL* est définie par les interprétations $T = (\triangle^{\mathcal{I}}, .^{\mathcal{I}})$. Dans T, $\triangle^{\mathcal{I}}$ est un ensemble non vide appelé le domaine d'interprétation ou l'univers d'interprétation, et $.^{\mathcal{I}}$ est une fonction d'interprétation qui affecte à chaque concept atomique A un ensemble $A^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}}$, et pour chaque rôle atomique R un ensemble $R^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$. Quelques définitions des extensions des descriptions induites par la fonction d'interprétation sont données dans le tableau 3.1.

Pour illustrer un de ces constructeurs présenté dans le tableau 3.1, en l'occurrence la conjonction, nous présentons l'exemple suivant. Si nous considérons deux concepts *Adventure* et *Sightseeing*, on a $Adventure \sqcap Sightseeing = Safari$, ce qui signifie que l'intersection entre les deux premiers concepts est le concept $Safari$ qui est composé de l'ensemble des individus qui sont à la fois instances du premier concept et du deuxième concept.

Les logiques de description sont utilisées, entre autres, dans le cadre du web sémantique et de l'ingénierie des connaissances pour la représentation des ontologies et la recherche de l'information et/ou l'inférence basée sur la logique.

3.2 Définition d'une ontologie

Selon Tom Gruber [39], "une ontologie est une spécification explicite d'une conceptualisation". Une définition plus détaillée d'une ontologie serait "une liste explicite et organisée de tous les termes, relations et objets qui constituent le schéma de représentation d'un domaine" [37].

En particulier, une ontologie permet de représenter les concepts d'un domaine et leurs relations, ou rôles, ainsi que la représentation d'un sous-ensemble d'objets appartenant à ces concepts et reliés par des liens instanciant les rôles génériques.

Les rôles sont habituellement orientés et se comportent comme des fonctions : un rôle fait correspondre aux instances sources d'un concept (domaine) les instances du concept cible (co-domaine). Par exemple, la figure 3.3 représente une vue **partielle** d'une onto-

Nom du constructeur	Syntaxe	Sémantique
concept racine	$\top$	$\triangle^{\mathcal{I}}$
concept vide	$\bot$	$\emptyset$
conjonction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjonction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
restriction universelle	$\forall R.C$	$\{a \in \triangle^{\mathcal{I}} \mid \forall b, (a,b) \in R^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}}\}$
restriction existentielle	$\exists R.C$	$\{a \in \triangle^{\mathcal{I}} \mid \exists b, (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$
équivalence	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$
restriction numérique au moins	$(\leqslant_n R)^{\mathcal{I}}$	$\{a \in \triangle^{\mathcal{I}} \mid \{b \mid (a,b) \in R^{\mathcal{I}}\}\| \leq n\}$
restriction numérique au plus	$(\geqslant_n R)^{\mathcal{I}}$	$\{a \in \triangle^{\mathcal{I}} \mid \{b \mid (a,b) \in R^{\mathcal{I}}\}\| \geq n\}$

Tableau 3.1 – Syntaxe et sémantique de quelques expressions de *DL*.

logie relative au domaine du voyage et de tourisme, appelée *Travel* [1].
Le graphe de la figure contient deux types de noeuds. Les noeuds rectangulaires représentent les classes (concepts du domaine) et les noeuds avec côtés arrondis représentent les propriétés (rôles) des concepts, alors que les arcs discontinus représentent les liens de domaine et de co-domaine, et les liens simples représentent la relation binaire *est-un* entre les concepts et entre les propriétés.

Dans cette figure, on peut distinguer des concepts tels que *Destination*, *Activity* et *Accommodation* et des rôles tels que *hasActivity* et *hasAccommodation*.

La figure 3.4, une capture d'écran de l'ontologie *Travel* visualisée via la plateforme *Protégé* [79], représente la hiérarchie de classes ainsi que quelques instances de classes avec leurs liens. La hiérarchie de classes est affichée sur la partie gauche de l'image dans laquelle on remarque que la classe *Hotel* et sa sous-classe *LuxuryHotel* sont sélectionnées. Les instances de ces deux classes sont affichées à droite de l'explorateur de classes. La sélection d'une instance permet d'afficher les autres individus auxquels cette instance est liée ainsi que le type de ces liens. Par exemple, la figure montre que l'instance *Clarion_StJames* est liée à l'instance *ThreeStarRating* via un lien de type *hasRating*. La figure 3.5, quant à elle, représente une capture d'écran des instances de l'ontologie *Travel* telles que visualisée par le plugiciel *OWL-DL Individuals* de *Protégé*. Les instances

1. http ://protege.stanford.edu/plugins/owl/owl-library/travel.owl.

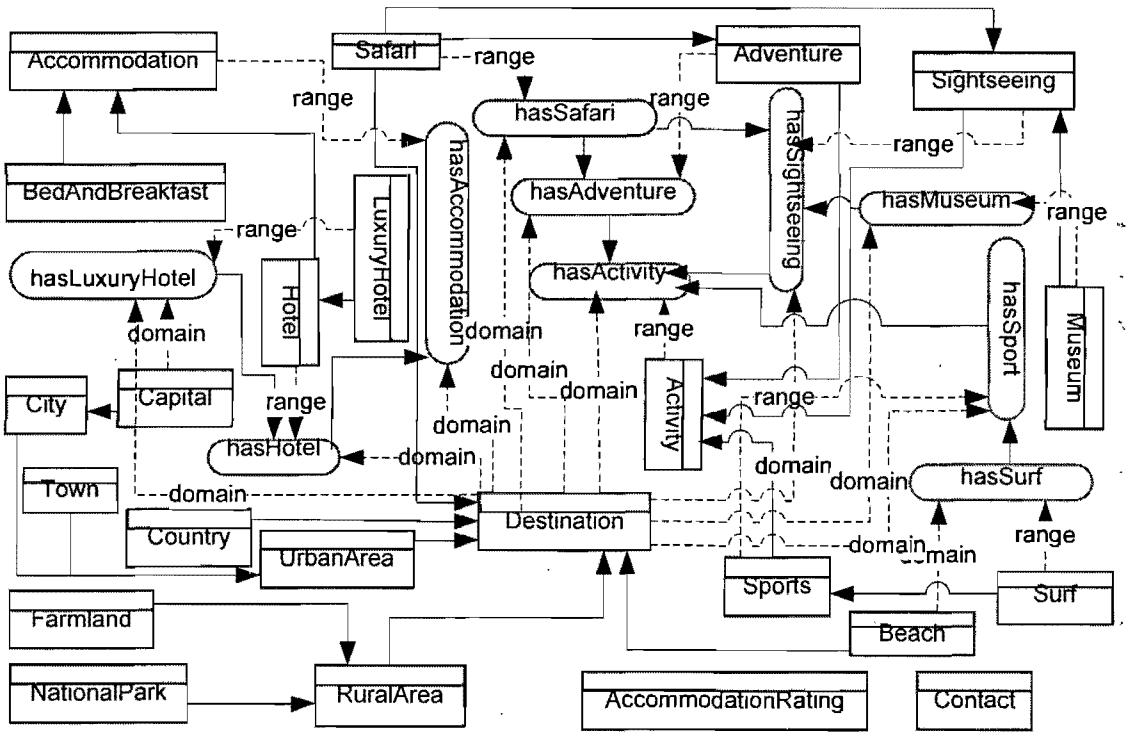


Figure 3.3 – Vue partielle de l'ontologie *Travel*.

sont regroupées par classe d'appartenance. Par exemple, les instances *Little_Buffalo*, *The_2_Campers* et *GUMIN_GUMIN_HOMESTEAD* sont des instances de la classe *Campground*.

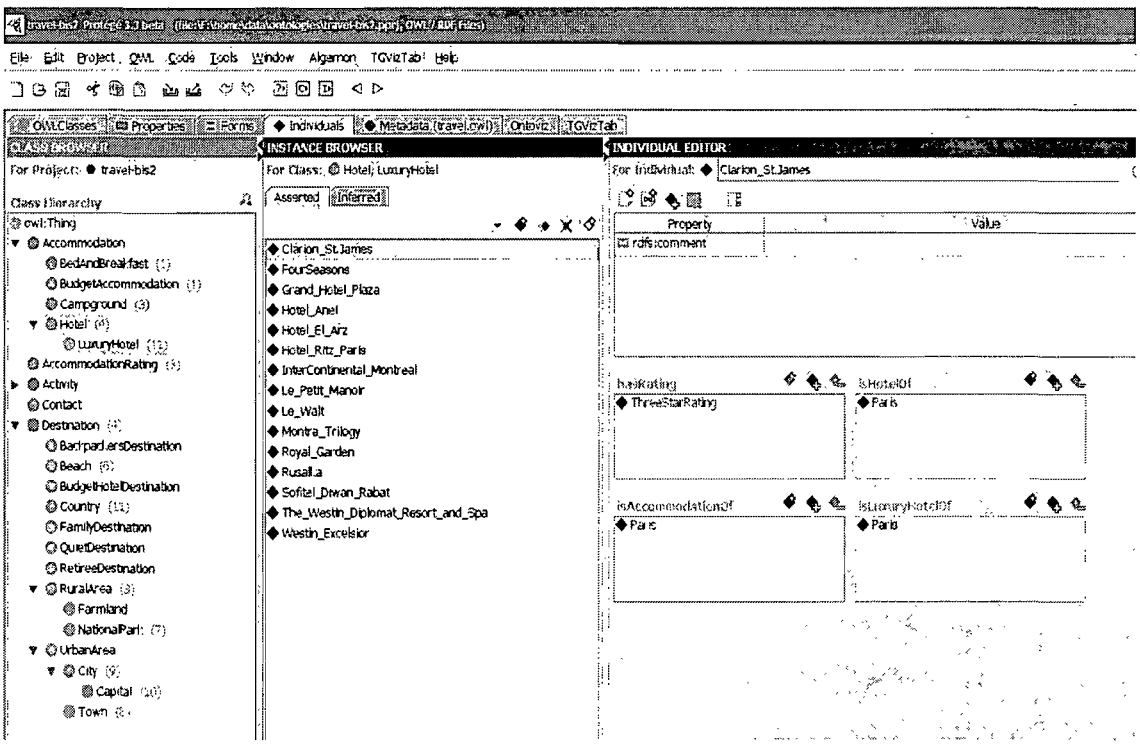


Figure 3.4 – L'ontologie *Travel* sur la plateforme *Protégé*.

Avec l'émergence du web sémantique [23], un intérêt croissant a été accordé aux ontologies, et des efforts de standardisation sont en cours dans le domaine des langages de description des ontologies (ex., OWL [51]).

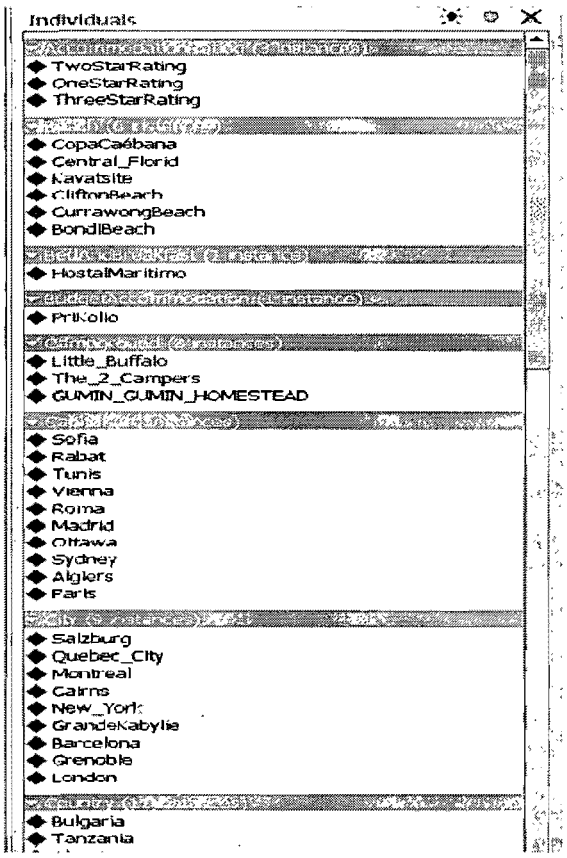


Figure 3.5 – L'ontologie *Travel* telle que visualisée via le plugiciel *OWL-DL* de *Protégé*.

3.3 OWL

OWL (*Ontology Web Language*) est un langage de représentation des connaissances d'un domaine spécifique utilisé dans le cadre du Web sémantique [51]. Fondé sur les principes des logiques de description (*DL*) [15], OWL dispose d'une syntaxe XML et permet d'exprimer des ontologies du Web sémantique sous forme de concepts et de rôles à partir d'un ensemble de connecteurs spécifiques au langage. Cela signifie qu'une ontologie Ω est exprimée en utilisant un langage dont les blocs constructifs sont un ensemble de concepts $\mathscr{C}_\Omega$, appelés aussi concepts atomiques, et d'un ensemble de rôles $\mathscr{R}_\Omega$. Les concepts et les rôles sont combinés pour former des descriptions plus complexes en utilisant un ensemble de constructeurs tels que l'union, l'intersection, la restriction de rôles, *etc.* Les instances, appelées individus ou objets, sont introduites comme un ensemble d'identifiants O_Ω (ou simplement O). Même si les langages de description d'ontologies fournissent des mécanismes de raisonnement tels que le calcul de la subsomption des descriptions, nous nous limitons dans l'usage de l'ontologie à l'utilisation stricte de ses fonctions de représentation. Ainsi, nous supposons disposer de la relation de généralité (*est-un*) entre les descriptions, c'est-à-dire, entre les concepts et entre les rôles dans Ω, notée $\sqsubseteq_\Omega$, et la relation de subsomption calculée $\sqsubseteq_\Omega$ qui est l'extension de la relation $\sqsubseteq_\Omega$ aux concepts et rôles de Ω. Il est à noter que nous allons utiliser $\sqsubset_\Omega$ comme notation pour la subsomption stricte ($S \sqsubset_\Omega S'$ si $S \sqsubseteq_\Omega S'$ et $S \neq S'$). Par exemple, de la même manière que *Hotel* est une sous-classe de *Accommodation*, *hasHotel* est un sous-rôle de *hasAcccommodation*. En outre, nous utilisons deux fonctions pour exprimer les connexions entre les rôles et les concepts qu'ils décrivent. Ainsi, étant donnée une relation $r \in \mathscr{R}_\Omega$, $dom(r)$ fournit les concepts les plus généraux qui ont le rôle r décrit, alors que $ran(r)$ fournit les concepts les plus généraux qui sont spécifiés dans l'ontologie comme concepts cible (co-domaine) de la relation r. Comme version restreinte, nous avons $ran(r,c)$ avec $c \in \mathscr{C}_\Omega$. Cette dernière fonction calcule les concepts du co-domaine associés au concept source c. De plus, comme $\sqsubseteq_\Omega$ est un ordre partiel, nous parlons de concepts prédécesseurs ($pred_\Omega(c)$) et de concepts successeurs ($succ_\Omega(c)$) d'un concept c dans la hiérarchie de concepts *est-un*. Par exemple, le concept *Accommodation* est

prédécesseur du concept *Hotel* (*Hotel* est successeur de *Accommodation*). De la même façon, nous définissons les successeurs d'un rôle ($pred_\Omega(r)$) et les prédécesseurs d'un rôle ($succ_\Omega(r)$) dans la hiérarchie de rôles *est-un*. Les descriptions qui ne figurent ni dans la relation de succession ni dans la relation de précédence sont incomparables, et cette relation sera notée $\perp$. Ainsi, les concepts *Hotel* et *City* sont incomparables. L'attribution de rôles-concepts est une relation ternaire $\rho \subseteq \mathscr{C} \times \mathscr{R} \times \mathscr{C}$ qui relie un rôle r aux classes c_1 et c_2 dont les instances sont, respectivement, origines et destination du lien de même type que r. De la même façon, nous définissons l'attribution de liens-objets comme étant une relation ternaire $\rho_o \subseteq \mathscr{O} \times \mathscr{R} \times \mathscr{O}$ qui relie un lien de type r aux objets o_1 et o_2.

Dans le tableau 3.2, nous définissons les différentes notations portant sur les ontologies, et que nous utilisons dans le cadre de la présente thèse.

Les spécifications d'un langage de description d'ontologie, tel que OWL, sont typiquement fournies avec une sémantique de dénotation abstraite qui requiert un domaine d'interprétation et une fonction d'interprétation. Cependant, pour nos besoins, nous limitons ce domaine à l'ensemble des instances O, ce qui signifie que la sémantique d'un concept c est l'ensemble de tous les individus qui sont explicitement spécifiés comme instances de c ou d'un de ses sous-concepts. De manière identique, un rôle r est interprété comme un ensemble de liens entre les individus de O. Ces sémantiques concrètes sont conjointement dénotées par $[]_\Omega$. Dans le reste de cette thèse, nous allons utiliser les termes tels que classe, objet et relation pour indiquer un concept, un individu et un rôle respectivement.

OWL est défini en trois sous-langages de plus en plus expressifs : OWL-Lite, OWL-DL et OWL-Full. Chacun de ces langages est conçu pour être utilisé par des communautés spécifiques suivant leurs besoins en pouvoir d'expression et selon la nature des contraintes recherchées. Le style le plus libre est appelé OWL-Full et est muni d'un pouvoir d'expression maximal. Par contre, il ne garantit pas la calculabilité. Dans ce langage, les trois parties de l'univers OWL sont identifiées par leurs contreparties RDF (un cadre d'application pour la description de ressources), à savoir les extensions de *rdfs :Resource*, *rdfs :Class* et *rdf :Property*. Dans OWL-Full, comme dans RDF, un élément de l'univers OWL peut à la fois être un individu et une classe ou, même être un

individu, une classe et une propriété. Dans OWL-DL, un style plus restrictif que OWL-Full, les trois parties diffèrent de leurs contreparties RDF et sont, en outre, disjointes. Le language OWL-DL sacrifie une part de son pouvoir d'expression au profit de la décidabilité de l'inférence. OWL-DL inclut tous les constructeurs offerts par OWL, mais ces derniers peuvent seulement être utilisés sous certaines restrictions. Par exemple, une classe peut être sous-classe de plusieurs classes différentes, alors qu'elle ne peut être une instance d'une autre classe. Le style le plus restrictif est OWL-Lite qui supporte les besoins basiques de classification hiérarchique et de système de contraintes simples. En effet, dans OWL-Lite, la cardinalité des restrictions sur les concepts et les rôles est limitée à 0 ou 1.

Dans le cadre de ce présent travail, nous sommes intéressés par OWL-DL qui implémente toutes les fonctionalités offertes par le formalisme des logiques de descriptions (*DL*) dont les bases ont été présentées dans la sous-section 3.1.

3.4 Discussion

La représentation des connaissances avec les ontologies est un processus qui vise à construire une image ou un modèle qui reflète notre perception d'un domaine. Ceci peut se limiter à la définition de l'ensemble des termes liés au domaine, ou aller plus loin en construisant une représentation structurée et hiérarchisée de ces termes avec les liens et relations qui leur sont associés. Une représentation riche et détaillée des connaissances permet, entres autres, aux machines (i) de raisonner avec un haut niveau d'abstraction, (ii) d'offrir des interprétations liées au domaine d'application considéré et (iii) d'éviter d'avoir toute ambiguïté.

Les systèmes de personnalisation peuvent exploiter les représentations de connaissances riches d'un domaine afin d'améliorer les prédictions et les suggestions. Étant donnée la complexité inhérente aux structures de ces connaissances, leur intégration soulèvera des questions et des problèmes aussi bien d'ordre théorique que pratique. Une démarche possible serait de définir un langage spécifique qui intègre les différentes entités des connaissances disponibles, au-dessus duquel des techniques de personnalisation

seront bâties. Naturellement, cette définition passera par l'élaboration d'une syntaxe et d'une sémantique appropriées aux éléments d'un tel langage.

Expression	Définition
Ω	ontologie du domaine
$\mathscr{O}_\Omega$	ensemble des objets de l'ontologie : $o_1, o_2, ..., o_l$
$\mathscr{C}_\Omega$	ensemble des noms des concepts de l'ontologie : $c_1, c_2, ..., c_m$
$\mathscr{R}_\Omega$	ensemble des noms des relations de l'ontologie : $r_1, r_2, ..., r_n$
$\sqsubseteq_\Omega$	relation de subsomption de concepts et de subsomption de relations dans Ω
$dom(r)$	concepts du domaine de définition de la relation r
$ran(r)$	concepts du co-domaine de la relation r
$ran(r, c)$	co-domaine de r lorsque son domaine est restreint à c
$pred_\Omega(c_i)$	ensemble des prédécesseurs du concept c_i dans Ω
$pred_\Omega(r_i)$	ensemble des relations plus générales que r_i dans Ω
$c_i \perp c_j$	c_i, c_j sont incomparables si c_i n'est ni parent ni enfant de c_j
$r_i \perp r_j$	les relations r_i et r_j sont incomparables si r_i n'est ni successeur ni prédécesseur de r_j

Tableau 3.2 – Définitions de base liées aux ontologies

Chapitre 4

Langage de motifs

Dans le chapitre 1, nous avons souligné que les approches et algorithmes de fouille des motifs séquentiels classiques ne prennent en considération que les identifiants des objets (AprioriSome[11], AprioriAll [11], PrefixSpan [60], SPADE [88], etc.) et qu'elles restent restrictives par rapport aux modes d'expression offerts par une ontologie, à savoir, la description d'un individu par trois types d'éléments : ses propriétés, ses divers liens à d'autres individus et l'ensemble des concepts (classes) dont il est instance. Sur ce point, les approches de fouille de motifs dits "généralisés" [76] constituent un premier pas pour l'intégration de connaissances ontologiques dans le processus de fouille car elles permettent d'extraire des motifs abstraits qui ne se limitent pas aux objets individuels, mais impliquent aussi des concepts génériques. Cependant, ces méthodes se limitent à la fouille de motifs en présence de catalogues hiérarchiques (taxonomies) et non d'ontologies complètes. De ce fait, elles ignorent des aspects primordiaux de la description des objets, tels que les relations inter-concepts qui font ressortir les dépendances sémantiques entre les concepts reflétant celles existantes au niveau objet.

Dans le présent chapitre, nous nous intéressons à des données qui complètent les possibilités d'expression des motifs avec certains éléments de la logique des prédicats. Nous présentons un nouveau langage de motifs qui prend en considération les connaissances additionnelles sur les objets du contenu. En clair, il s'agit d'intégrer le maximum de connaissances ontologiques du domaine. Les motifs résultants sont des graphes orientés composés de séquences de concepts complétées par les relations inter-concepts ontologiques à différents niveaux d'abstraction [3–5]. Afin de pouvoir distinguer entre les aspects purement structuraux des motifs et la signification de ces structures, le langage est muni d'un ensemble de constructeurs syntaxiques dotés d'une sémantique précise.

4.1 Connaissances ontologiques et fouille de motifs séquentiels

La découverte de motifs comportementaux dans les logs des applications Web facilite l'adaptation de ces applications, aussi bien du point de vue du contenu servi que de la structuration de celui-ci en pages Web [62]. Un exemple typique du bénéfice apporté par ces motifs est leur application dans les systèmes de recommandation : en prédisant les consultations les plus probables sur la base d'un ensemble de consultations déjà effectuées au préalable et des motifs extraits des sessions passées. Un tel système est capable de suggérer à son utilisateur le ou les "meilleurs" choix pour une prochaine consultation. Ceci constitue un formidable champ d'application pour les méthodes de découverte de motifs fréquents et de règles d'association qui a déjà donné naissance à toute une branche de la fouille de données appelée *Web usage mining*. Typiquement, les méthodes sous-jacentes vont considérer les collections de pages consultées par un utilisateur afin d'en extraire les pages qui apparaissent fréquemment ensemble, et qui par la suite seront interprétées et des conclusions en seront tirées sur les habitudes des utilisateurs (*access patterns*) et sur l'adéquation des informations servies par l'application.

Dans notre manière de poser le problème, les données initiales sont des séquences d'objets prélevées des logs d'une application Web. Afin d'intégrer les connaissances ontologiques, chaque séquence est enrichie avec les liens qui existent entre deux de ses objets et qui vont dans le sens de l'ordre temporel de la séquence. Le résultat est donc un graphe orienté sans cycle (appelé séquence d'objets étendue) dont les arcs sont étiquetés par des noms de relations de l'ontologie ou par une relation spéciale, celle exprimant l'ordre temporel (la séquence). C'est à partir d'un ensemble de tels graphes que la recherche de sous-graphes fréquents (appelés aussi motifs fréquents) va s'opérer.

À notre connaissance, la fouille de motifs à partir de données ainsi décrites — combinant des aspects de plusieurs approches décrites ci-dessus — n'a pas encore été abordée dans la littérature. Pour cette raison, le but de ce chapitre est de présenter les fondements de l'approche, en particulier, un langage de description de motifs au-dessus du langage de l'ontologie (ici, OWL). Une relation de généralité entre motifs est également pro-

posée, et elle est basée sur l'interprétation de ceux-ci (motifs) en terme de séquences d'objets enrichies "couvertes" par un motif. Cette relation permet d'organiser l'espace de tous les motifs dans une structure d'ordre partiel qui facilite le parcours à la recherche du sous-ensemble des motifs fréquents. À ce niveau, deux possibilités de construction de l'espace des motifs fréquents nous sont offertes : (i) par généralisation de motifs, et (ii) par spécialisation de motifs. La construction par généralisation consiste à démarrer des séquences étendues et à rechercher les motifs qui les généralisent. La construction par spécialisation consiste en un parcours du haut vers le bas de l'espace des motifs en passant à chaque fois d'un motif à un autre motif plus spécifique.

Afin de rendre possible une stratégie de parcours qui profite de l'anti-monotonie de la propriété de fréquence par rapport à l'ordre partiel des motifs [1] (la même que celle utilisée par *Apriori* [75]), une partition en niveaux de cette structure est étudiée. Celle-ci est enrichie avec un ensemble d'opérations qui étendent un motif (spécialisation) et qui sont des transformations élémentaires. Ces opérations permettent de traverser les niveaux en suivant la relation de précédence qui est la réduction transitive de la relation de généralité entre motifs.

Nos caractérisations des structures sous-jacentes à la notion de séquence étendue sont complétées par la description d'une méthode algorithmique d'extraction de tous les motifs fréquents à partir d'un ensemble de données. Notre méthode, à l'instar d'*Apriori*, applique une stratégie de parcours descendante, en largeur d'abord et par niveau, en utilisant les résultats de la recherche sur un niveau pour alimenter la génération de candidats au niveau suivant.

Le travail décrit dans ce chapitre ignore, de façon délibérée, les questions liées à une recherche dite "verticale" dans l'espace des motifs, à l'image de celle incorporée dans les algorithmes *CHARM* [90], *Closet* [59], *PrefixSpan* et *FP-growth* [41, 60, 61], etc. De même, les notions de motifs fermés [77] ou maximaux [82] dans ce contexte ont été réservées pour des développements ultérieurs.

Le reste de ce chapitre est organisé comme suit. Dans la section 4.2, nous définissons

1. Anti-monotonicité de la fréquence de motifs : lorsqu'un motif est fréquent, tous ses sous-motifs le sont aussi.

les motifs et présentons les différentes notions et principes derrière ce nouveau langage tels que les langages de description des données et des motifs. Le chapitre se termine par la section 4.3 où une conclusion est présentée.

4.2 Langage de description de motifs

Dans notre méthodologie de développement d'une approche pour la fouille de motifs fréquents, nous proposons deux langages : le premier pour la description des données et le deuxième pour la description des motifs. Comme notre recherche de motifs porte sur plusieurs séquences correspondant aux sessions des différents utilisateurs d'un site Web, les données de départ sont composées d'un ensemble de séquences d'objets.

4.2.1 Langage de description des données

Une séquence d'objets (ex. items visités/consultés dans un certain ordre durant une période de temps) est définie comme étant une liste ordonnée d'items (objets). La figure 4.1 représente la séquence s_1 composée des objets *France*, *Grenoble*, *Paris*, *Louvre* et *Ritz* où l'ordre de consultation est indiqué par des flèches en pointillés. Les séquences d'objets sont facilement traduites en identifiants d'objets de l'ontologie. Ainsi, une séquence de données $s_{\mathcal{O}_\Omega}$ appartient à l'univers des séquences $O^\omega = (\mathcal{O}_\Omega \times \mathcal{O}_\Omega \times ...)$, où $\mathcal{O}_\Omega$ est l'ensemble de tous les objets de l'ontologie du domaine considéré.

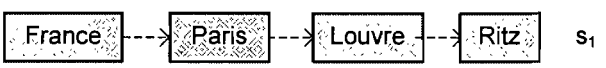


Figure 4.1 – Séquence d'objets.

Même si les motifs peuvent être définis au niveau objets, ils ont un intérêt limité. En effet, en présence d'un grand nombre d'objets $\mathcal{O}_\Omega$, les chances de retrouver des combinaisons d'objets fréquents est relativement faible. En revanche, passer au niveau classe augmente considérablement ces chances. À cette fin, les séquences d'objets initiales sont

en premier lieu étendues en incorporant tous les liens valides de l'ontologie. Pour des raisons de concision, nous nous limitons à l'insertion des liens qui sont "co-linéaires" avec l'ordre des objets dans la séquence d'objets de départ (c'est-à-dire, l'objet source du lien est avant l'objet destination). À titre d'exemple, la figure 4.2 est une illustration de l'augmentation de la séquence de la figure 4.1 par des liens puisés de l'ontologie *Travel*. Ainsi, les objets *Paris* et *Louvre* sont liés par le lien *hasMuseum* pour indiquer que le musée *Louvre* se trouve dans la ville *Paris*. De même, les objets *Paris* et *Ritz* sont liés par le lien *hasLuxuryHotel* pour indiquer que *Paris* est la ville où se trouve l'hôtel *Ritz*.

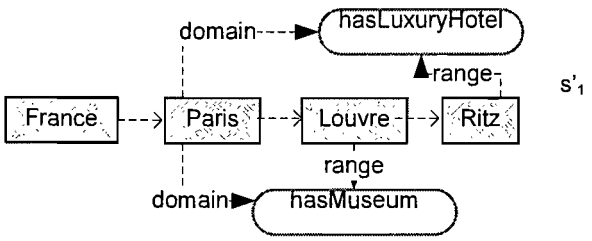


Figure 4.2 – Séquence d'objets étendue.

4.2.2 Syntaxe des séquences d'objets étendues

Chaque séquence d'objets étendue s a une structure de graphe dont les sommets et les arcs sont étiquetés. Les sommets représentent des objets et les arcs représentent les liens entre ces objets dans l'ontologie Ω. Ainsi, les séquences d'objets de départ désignées par l'ensemble $\mathscr{D}$ sont transformées en séquences d'objets étendues dont l'ensemble est désigné par $\mathscr{D}_\Omega$ avec $\mathscr{D}_\Omega \subseteq \Delta_\Omega$, où Δ_Ω est l'univers de toutes les séquences d'objets étendues.

Une séquence d'objets étendue se base principalement sur la relation d'interprétation des relations d'une ontologie qui est définie ci-dessous conjointement avec la définition de l'interprétation de concepts.

Considérons un ensemble d'objets $\mathscr{O}_\Omega$, l'ensemble des concepts et des relations de Ω qui sont dénotés respectivement par $\mathscr{C}_\Omega$ et $\mathscr{R}_\Omega$, et deux fonctions :

- $[\,]_{\mathscr{C}_\Omega} : \mathscr{C}_\Omega \rightarrow E(\mathscr{O}_\Omega)$ [2] tel que : $c \in \mathscr{C}_\Omega$ et $o \in \mathscr{O}_\Omega$ on a $o \in [c]_{\mathscr{C}_\Omega}$ seulement si o est instance de c. Par exemple, l'objet $Paris \in [Capital]_{\mathscr{C}_\Omega}$.
- $[\,]_{\mathscr{R}_\Omega} : \mathscr{R}_\Omega \rightarrow E(\rho_{\mathscr{O}})$ [3] tel que : si $(o_1, r, o_2) \in [r]_{\mathscr{R}_\Omega}$ alors $o_1 \in [c_1]_{\mathscr{C}_\Omega}$, $o_2 \in [c_2]_{\mathscr{C}_\Omega}$, $c_1 \in dom(r)$, et $c_2 \in ran(r, c_1)$. Par exemple, le triplet (*Paris*, *hasMuseum*, *Louvre*) $\in$ $[hasMuseum]_{\mathscr{R}_\Omega}$.

Dans la suite, la notation $[\,]_\Omega$ sera d'une part utilisée à la place des notations $[\,]_{\mathscr{C}_\Omega}$ et $[\,]_{\mathscr{R}_\Omega}$ et c'est le contexte d'utilisation qui déterminera la relation d'interprétation exprimée, suivant qu'il s'agisse d'un concept ou d'une relation. D'autre part, nous considérons seulement l'interprétation effective des concepts et des relations. Cela signifie que l'interprétation d'un concept (resp. d'une relation) se ramènera aux objets (resp. couples d'objets) présents dans l'ontologie du domaine considéré.

Ainsi, la définition formelle de la syntaxe abstraite de notre langage de données se présente comme suit :

Définition 4.2.1. ***Séquence d'objets étendue***

Une séquence d'objets étendue est une paire $s = (\varsigma, \theta)$ où $\varsigma \in O^\omega$ est une séquence d'objets (notée $s.\varsigma$) et $\theta \in E(\rho_{\mathscr{O}})$ est un ensemble de liens (noté $s.\theta$). Pour des raisons de concision, les triplets de $\rho_{\mathscr{O}}$ seront simplifiés de la forme (o_d, r, o_r) à $r(l,m)$ où l et m sont des nombres naturels qui dénotent, respectivement, les positions de o_d et o_r dans $s.\varsigma$. De plus, les contraintes suivantes sont à respecter :

1. *$\forall r(l,m) \in s.\theta$, $l \leq m$, où les relations sont co-linéaires avec l'ordre des objets de ς,*
2. *$\forall r(j,k) \in s.\theta$: $(s.\varsigma[j], s.\varsigma[k]) \in [r]_\Omega$, où $s.\varsigma[x]$ fait référence à l'objet de s se trouvant à la position x.*

La figure 4.3 représente un exemple de transformation de la séquence d'objets s_1 en la séquence étendue s_1'. Les deux liens *hasLuxuryHotel* et *hasMuseum* sont ajoutés à s_1

2. $E(X)$ représente l'ensemble des parties de l'ensemble X
3. $\rho_{\mathscr{O}}$ est la relation ternaire d'attribution de liens-objets (voir le chapitre 3)

entre les objets se trouvant respectivement aux positions (2,4) et (2,3). Ainsi, s'_1 sera représentée par $(\langle France, Paris, Louvre, Ritz\rangle, \{hasLuxuryHotel(2,4), hasMuseum(2,3)\})$.

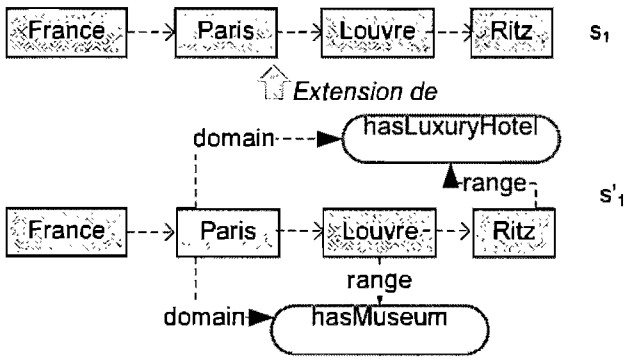


Figure 4.3 – Exemple de transformation de séquence d'objets.

4.2.3 Algorithme de transformation de séquences d'objets

Comme mentionné plus haut, il s'agit de transformer une séquence d'objets en une structure plus complexe composée d'objets et de liens puisés de l'ontologie du domaine. Cette transformation est expliquée dans l'algorithme 2 où les différents liens de l'ontologie existant entre les objets de la séquence, et qui sont co-linéaires avec l'ordre des objets de cette dernière, sont ajoutés à la séquence d'objets (*cf.* lignes 12 à 16).

4.2.4 Syntaxe des motifs

Soient Δ l'univers de toutes les séquences d'objets possibles Δ_Ω l'ensemble des séquences étendues associées. Une séquence de classes étendue, désignée aussi par le terme motif, est composée d'une liste de classes, où deux classes voisines peuvent être comparables, et d'un ensemble de triplets représentant les relations de l'ontologie qui relient les concepts de la séquence. L'ensemble de tous les motifs pouvant être produits à partir de l'ontologie Ω sera désigné par Γ_Ω. Formellement, la syntaxe abstraite de notre langage de motifs est définie comme suit :

Algorithm 2 ISEQTRANS : Transformation de séquences d'objets en séquences d'objets étendues

Entrée
$\mathscr{D}_\Omega \leftarrow \emptyset$; ▷ ensemble vide de séquences étendues
$\mathscr{R}_\Omega$;
$\mathscr{D} = \{s_1^{\mathscr{O}}, s_2^{\mathscr{O}}, .., s_n^{\mathscr{O}}\}$ ▷ ensemble des séquences d'objets non vides

Sortie
$\mathscr{D}_\Omega$, ▷ ensemble des séquences d'objets étendues

Méthode
for $i = 1$ à n **do** ▷ Pour chaque séquence ajouter les liens de l'ontologie qui sont co-linéaires avec l'ordre des objets dans cette séquence
    $s.\theta \leftarrow \emptyset$;
    $s.\varsigma \leftarrow s_i^{\mathscr{O}}$;
    for $j = 1$ à $|s.\varsigma|$ **do**
        for $k = j$ à $|s.\varsigma|$ **do**
            for $\forall r \in \mathscr{R}_\Omega$ **tel que** $r(s.\varsigma[j], s.\varsigma[k]) \in \rho_{\mathscr{O}}$ **do**
                $s.\theta \leftarrow s.\theta \cup \{r(j,k)\}$
            end for
        end for
    end for
    $\mathscr{D}_\Omega \leftarrow \mathscr{D}_\Omega \cup \{s\}$
end for
return $\mathscr{D}_\Omega$;

Définition 4.2.2. ***Séquence de classes étendue (motif)***

Une séquence de classes étendue est une paire $S = (\varsigma, \theta)$ *avec :*

- $\varsigma \in C^{\omega}$ [4], $\varsigma = \langle c_i \rangle_{i=1..n}$ *une séquence de classes. Dans la suite de ce document, nous allons souvent désigner* c_i *par* $S.\varsigma[i]$,
- $\theta \in E(\rho)$ *un ensemble de relations. Les éléments de* $S.\theta$ *désignés par* (c_d, r, c_r), *seront ultérieurement désignés par* $r(l,m)$ *où* l *et* m *sont des entiers indiquant les positions des classes* c_d *et* c_r *dans la séquence* $S.\varsigma$ *et* $r \in \mathscr{R}_\Omega$ *une relation de l'ontologie* Ω. *Chaque élément* $r(l,m)$ *de* $S.\theta$ *satisfait les conditions suivantes :*
  - $l \leq m$,
  - $\exists\, c \in dom(r)$ *tel que* $c_l \sqsubseteq_\Omega c$ *ce qui sera désigné par* $c_l \ll dom(r)$,
  - $\exists\, c \in ran(r, c_l)$ *tel que* $c_m \sqsubseteq_\Omega c$ *ce qui sera désigné par* $c_m \ll ran(r, c_l)$.

À titre illustratif, la formulation textuelle de la séquence de classes étendue S de la figure 4.4 est comme suit : $S = (\langle Capital, LuxuryHotel, Museum \rangle, \{hasLuxuryHotel(1,2), hasMuseum(1,3)\})$. Cette séquence de classes étendue est composée des classes *Capital*, *LuxuryHotel*, *Museum*, de la relation *hasLuxuryHotel* liant les classes *Capital* et *LuxuryHotel*, et de la relation *hasMuseum* qui lie la classe *Capital* avec la classe *Museum*.

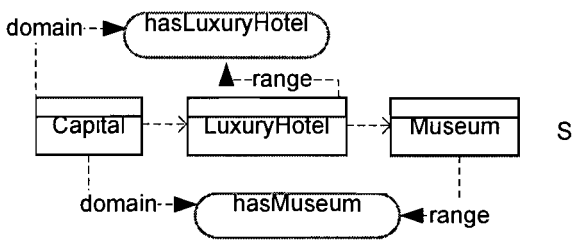


Figure 4.4 – Séquence de classes étendue.

Remarque 4.2.1. *Soit* c *un concept de* $\mathscr{C}_\Omega$ *et* r *une relation de* $\mathscr{R}_\Omega$. *L'expression* $\exists\, c' \in ran(r)$ *tel que* $c \sqsubseteq_\Omega c'$ *sera désignée par* $c \ll ran(r)$.

4. $C^{\omega} = (\mathscr{C}_\Omega \times \mathscr{C}_\Omega \times ...)$: ensemble de toutes les séquences de concepts qui peuvent être construites à partir de $\mathscr{C}_\Omega$.

Remarque 4.2.2. *L'espace* Γ_Ω *représente l'ensemble des motifs pouvant être construits à partir de* Ω*. Cet espace est muni du motif vide* $(\langle\rangle, \{\})$*, qui sera aussi représenté par* $\emptyset_{\Gamma_\Omega}$.

4.2.5 Sémantique des motifs

Un motif est interprété par un ensemble de séquences d'objets étendues. L'interprétation se base sur deux éléments : (i) le domaine d'interprétation constitué de toutes les séquences d'objets étendues, et (ii) la fonction d'interprétation qui associe à chaque motif un ensemble d'éléments du domaine d'interprétation.

La fonction qui associe à un motif l'ensemble de toutes les séquences d'objets étendues qui sont "résumées" par ce motif est basée sur l'intuition suivante :

En premier lieu, la structure de la séquence d'objets étendue est reflétée dans le motif de manière réduite. Une telle réduction signifie que ce ne sont pas tous les objets qui ont besoin d'être représentés dans le motif, quelques-uns peuvent simplement être écartés à cause de leur non pertinence. De façon similaire, les liens relationnels de la séquence d'objets sont représentés par des relations de l'ontologie ou simplement écartés. De plus, à chaque classe du motif doit correspondre un objet de la séquence d'objets étendue tout en respectant l'ordre des classes dans le motif. Il est aussi à noter qu'à chaque relation du motif correspond un lien dans la séquence d'objets étendue qui lie les objets associés aux classes de la relation en question.

De manière plus rigoureuse, nous exprimons l'instanciation entre une séquence et un motif par l'existence d'une correspondance partielle entre la séquence d'objets étendue et le motif tel que :

- un concept est mis en correspondance avec seulement un objet qui lui est instance,
- l'ordre dans le motif est préservé, c'est-à-dire, si la correspondance est définie pour deux concepts, alors leurs images dans la séquence d'objets étendue respectent nécessairement le même ordre,
- à chaque classe est associée exactement une seule image (un seul objet) dans la séquence d'objets étendue,

- à chaque relation du motif est associée exactement une seule image (un lien) –qui lui est instance– dans la séquence d'objets étendue. De plus, les classes adjacentes de la relation sont respectivement associées avec les objets adjacents du lien.

Il est évident qu'une telle association entre les éléments de la séquence d'objets étendue et les éléments du motif n'est pas nécessairement unique.

La relation d'instanciation de motif se base sur la relation d'interprétation des concepts et des relations présentée dans la section 4.2.2, et est formellement définie comme suit :

Définition 4.2.3. ***Relation d'instanciation***

Soient s une séquence d'objets étendue de Δ_Ω *et S une séquence de classes étendue de* Γ_Ω. *On dit que s est instance de S, dénotée par* $s \lhd S$, *s'il existe une sous-structure de s qui correspond homomorphiquement à S. En clair,* $s \lhd S$ *s'il existe une fonction injective monotone croissante* ϕ : $[1..|S.\varsigma|] \rightarrow [1..|s.\varsigma|]$ *tel que :*

- $\forall i \in [1..|S.\varsigma|], s.\varsigma[\phi(i)] \in [S.\varsigma[i]]_\Omega$,
- $\forall r'(i_1, i_2) \in S.\theta, \exists\ (j_1, j_2) \in ([1..|s.\varsigma|])^2$ *tels que* $\phi(i_1) = j_1, \phi(i_2) = j_2$, *et* $\exists\ r(j_1, j_2) \in s.\theta$, *où* $r \sqsubseteq_\Omega r'$ *et* $(s.\varsigma[j_1], s.\varsigma[j_2]) \in [r'(i_1, i_2)]_\Omega$.

Remarque 4.2.3. *Pour toute séquence d'objets étendue s de* Δ_Ω, *nous avons* $s \lhd \emptyset_{\Gamma_\Omega}$.

La figure 4.5 représente un exemple de mise en correspondance entre une séquence de classes étendue et une de ses instances (séquence d'objets étendue). Comme on peut le constater, des objets peuvent ne correspondre à aucune classe. En effet, dans la séquence S', la classe *Destination* est mise en correspondance avec l'objet *Paris* et non avec l'objet *France* cela est dû au fait que *France* n'est pas lié avec un autre objet dans s'_1 par une relation identique ou plus spécifique que la relation *hasAccommodation* qui part de *Destination*.

Dans la suite, nous présentons le processus de mise en correspondance par un algorithme qui teste l'instanciation d'un motif par une séquence d'objets étendue. Ceci

va nous permettre d'associer à un motif les séquences d'objets étendue résumées par ce motif.

De la figure 4.5, le motif S' préserve la structure de la séquence d'objets étendue s'_1 alors que le motif S ne préserve pas la structure de s'_1. Les arcs pointillés gris représentent la mise en correspondance des éléments des deux séquences.

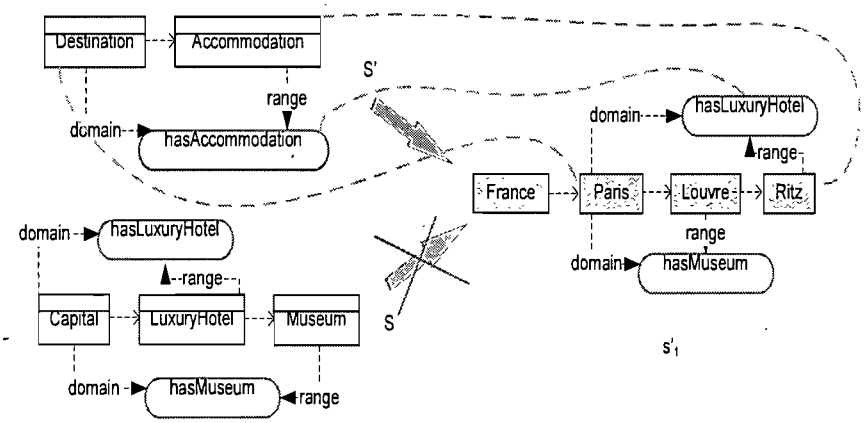


Figure 4.5 – Exemple d'une correspondance avec préservation de la structure.

4.2.5.1 Test d'instanciation

Dans cette section, nous présentons le test d'instanciation qui consiste à rechercher un homomorphisme de graphes entre un motif et une séquence d'objets étendue.

Un algorithme qui suit la définition 4.2.3 est développé. Cet algorithme se base sur le test des différents appariements possibles pour la mise en correspondance d'un motif avec une séquence d'objets étendue. Pour construire une nouvelle configuration, un ensemble d'objets dont la taille correspond au nombre de concepts du motif à apparier, sont tirés de l'ensemble de tous les objets de la séquence d'objets considérée. Par la suite, ces objets sont ordonnés suivant leur position dans la séquence du plus petit au plus grand. Maintenant, il est possible de commencer le processus d'appariement.

Dans chaque appariement, les classes sont alignées à leurs instances (objets) alors que les relations doivent être supportées par un lien ayant le même libellé. De plus, un

tel lien doit *(i)* démarrer de l'objet appartenant à l'image de la classe source de la relation, et *(ii)* aboutir à l'objet de l'image de la classe destination de la même relation.

Afin de réduire le nombre d'opérations à effectuer, la procédure *Inst()* regroupe ces appariements par préfixe. Ainsi, pour un ensemble d'appariements qui ont le même préfixe, les opérations sur le préfixe seront effectuées une seule fois.

La mise en correspondance entre un motif et une séquence d'objets étendue est un processus potentiellement coûteux. La raison est que plusieurs tentatives à différentes places dans s pourront être nécessaires avant de retrouver une image appropriée de c (*resp.* de r).

Étant donné que pour un concept (resp. une relation) plusieurs images potentielles peuvent être retrouvées dans la séquence d'objets étendue, l'algorithme devrait considérer différentes combinaisons pour retrouver l'image adéquate. Il est à souligner que toutes les images sont considérées comme étant candidates tant que l'algorithme n'est pas terminé en confirmant l'instanciation du motif par la séquence d'objets en question. Ainsi, l'image candidate d'un concept (resp. d'une relation) n'est que provisoire et peut être remise en cause à n'importe quel moment lors du test.

Pour rechercher les images des concepts (resp. des relations), la stratégie la plus simple consiste à parcourir les deux séquences comme suit. Au fur et à mesure qu'on avance dans la séquence de classes étendue, et qu'on rencontre un concept (resp. une relation), on cherche son image dans la séquence d'objets étendue. On s'arrête lorsqu'une image est retrouvée pour tout concept (resp. relation) ou qu'on arrive à la fin de la séquence d'objets étendue sans avoir retrouvé une image pour le concept courant (resp. la relation courante) du motif.

En procédant ainsi, on sera amené à passer du test d'un concept à un autre concept qui n'est pas systématiquement son voisin immédiat, ce qui compliquera davantage le test d'instanciation. En effet, lorsqu'une relation est rencontrée, non seulement on doit avoir à disposition l'image du concept courant mais aussi l'image du concept qui est destination de cette relation. Ceci peut faire oublier des concepts se trouvant entre les concepts source et destination d'une relation donnée. Avec des sauts éventuels dans les

deux séquences, la tâche de test devient complexe.

Afin de réduire cette complexité, nous apportons une optimisation au processus, laquelle consiste à ne traiter les relations qu'après avoir retrouvé les images des concepts. Autrement dit, il s'agit de rechercher l'image d'une relation une fois les images de tous les concepts et relations se trouvant avant le concept destination de la relation en question sont retrouvées.

Algorithm 3 Procédure de test d'instanciation

```
1:  procedure INST(S : motif ; s : séquence d'objets étendue)
2:      if |S.ς| > |s.ς| or |S.θ| > |s.θ| then
3:          return false;
4:      end if
5:      imgsPot ;          ▷ Tableau de listes contenant les images potentielles de chaque
    concept
6:      imgsPot ← TROUVERIMAGESPOTENTIELLES(S,s);
7:      i ← 1 ;
8:      while i ≤ |S.ς| do
9:          if imgsPot[i].next? == false then
10:             if i == 1 then
11:                 return false;          ▷ On n'a pas pu apparier le premier concept
12:             else
13:                 imgsPot[i].toFirst ; ▷ Pointer vers la première position de imgsPot[i]
14:                 i − − ;
15:                 break ;
16:             end if
17:         end if
18:         posImagCrt ← imgsPot[i].next() ;
19:         if i > 1 and posImagCrt < imgsPot[i − 1].current() then
20:             break ;
21:         end if
22:         if RELATIONSCORRESPONDENT(S,s,i,imgsPot,posImagCrt) then
23:             i + + ;
24:         end if
25:     end while
26:     return true;
27: end procedure
```

Techniquement parlant, le pseudo-code du test d'instanciation, présenté dans l'algorithme 3, commence par l'énumération des images potentielles de chaque concept de

Algorithm 4 Procédure qui vérifie s'il y a correspondance de relations

1: **procedure** RELATIONSCORRESPONDENT(S : motif ; s : séquence d'objets étendue, i : position d'un concept dans S, *imgsPot* : images potentielles, *posImagCrt* : position de l'image du concept de la position i)
2: **for all** $r(k,i)$ **in** $S.\theta$ **do**
3: **if** $\nexists r'(imgsPot[k].current(), posImagCrt) \in s.\theta$ et $r' \sqsubseteq_\Omega r$ **then**
4: **return** $false$;
5: **end if**
6: **end for**
7: **return** $true$;
8: **end procedure**

Algorithm 5 Procédure de recherche d'images potentielles

1: **procedure** TROUVERIMAGESPOTENTIELLES(S : motif ; s : séquence d'objets étendue)
2: **for** $i = 1 ; i \leq |s.\varsigma| ; i++$ **do**
3: **for** $j = 1 ; j \leq |S.\varsigma| ; j++$ **do**
4: **if** $(s.\varsigma[i] \in [S.\varsigma[j]]_\Omega)$ et $\forall r(i,k) \in S.\theta, \exists r'(j,k') \in s.\theta$: $r(i,k) \sqsubseteq_\Omega r'(j,k')$ et $\forall r(k,i) \in S.\theta, \exists r'(k',j) \in s.\theta\, r(k,i) \sqsubseteq_\Omega r'(k',j)$ et $(|s.\varsigma| - j \geq |S.\varsigma| - i)$ **then**
5: $imgsPotTemp[j].addElement(i)$;
6: **end if**
7: **end for**
8: **end for**
9: **return** *imgsPotTemp*;
10: **end procedure**

S en appelant la procédure dont le pseudo-code est décrit par l'algorithme 5. Ce dernier parcourt s une seule fois. Lors de ce parcours, une image d'un concept n'est pas prise en considération si sa position dans s n'offre aucune chance au reste des concepts de S d'avoir une image potentielle. Concrètement, un objet o est ajouté à la liste des images potentielles d'un concept c, s'il est instance de ce concept, qu'il supporte dans s les mêmes relations entrantes et sortantes que celles de c dans S. De plus, il faut que le nombre d'objets se trouvant après o soit supérieur ou égal au nombre de concepts qui suivent c dans S (voir les lignes 4 à 6 de l'algorithme 5).

Une fois que les images potentielles sont déterminées, toutes les combinaisons possibles sont testées une à une. Ce test se déroule comme suit. On itère sur tous les concepts de S, et pour chaque concept on teste les images potentielles dans l'ordre croissant (voir les lignes 8 à 25).

Pour chaque concept, on s'assure qu'il existe au moins une image potentielle qui lui est associée et qu'elle n'a pas encore été traitée. Dans le cas contraire, on vérifie si c'est le premier concept. Si ce n'est pas le cas, le suffixe courant est invalidé en remettant en cause l'image du concept qui précède le concept courant (voir les lignes 9 à 17).

Dans le cas où il existe encore des images potentielles associées au concept courant et qui ne sont pas encore traitées, on prend l'image suivante et les images des relations entrantes sont recherchées. Le pseudo-code de l'algorithme 4 représente la procédure qui vérifie la correspondance de relations. Ainsi, une image est recherchée pour chaque relation qui aboutit au concept courant c dans S. S'il existe au moins une relation pour laquelle il n'existe pas d'image à la position courante, l'image du concept courant est remise en cause, et une autre image potentielle sera testée à la prochaine itération. Par contre, si pour toutes les relations entrantes une image est retrouvée, l'algorithme avance dans S.

Il est à remarquer que l'algorithme arrête de tester la combinaison courante si le préfixe est invalidé. En effet, dès que l'image d'un concept n'est pas validée, l'algorithme abandonne l'exploration du préfixe courant et revient en arrière pour tester un nouveau préfixe. L'algorithme se termine si la séquence d'objets étendue est entièrement parcourue en y parvenant à retrouver une image pour chaque concept et relation, ou que l'on

se (re)trouve au niveau du premier concept et que toutes les images potentielles (si elles existent) ont été testées.

4.2.5.2 Interprétation et relation de généralité

Alors que la relation d'instanciation permet de déterminer si une séquence d'objets étendue est représentée par une séquence de classes étendue, la fonction d'interprétation est l'extension de la relation d'instanciation à toutes les séquences d'objets étendues qui sont couvertes pas une séquence de classes étendue donnée.

En se basant sur la relation d'instanciation de motifs et en notant la fonction d'interprétation par $[\,]_{\Gamma_\Omega} : \Gamma_\Omega \rightarrow E(\Delta_\Omega)$, nous avons ce qui suit :

Définition 4.2.4. ***Fonction d'interprétation***

L'interprétation de la séquence de classes étendue S, dénotée par $[S]_{\Gamma_\Omega}$*, est constitué des séquences d'objets étendues s de* Δ_Ω *tel que* $s \lhd S$ *et prend la forme suivante :* $[S]_{\Gamma_\Omega} = \{s \in \Delta_\Omega | s \lhd S\}$.

À titre d'exemple, l'ensemble d'interprétation de la séquence de classes étendue S' de la figure 4.6 inclut les séquences d'objets étendues représentées dans la même figure. En effet, les séquences s'_1, s'_2 et s'_3 sont instances de S'.

En se basant sur la notion de sémantique des motifs et la relation d'interprétation, nous définissons la relation de généralité entre motifs. Cette dernière (relation de généralité) nous permet d'ordonner l'espace des motifs en hiérarchies. Formellement nous avons ce qui suit :

Définition 4.2.5. ***Relation de généralité***

Soient S_1 *et* S_2 *deux séquences de classes étendues de* Γ_Ω*, on dit que* S_1 *est plus générale que* S_2*, qu'on note par* $S_2 \leq S_1$ *ou encore par* $S_1 \geq S_2$*, si et seulement si*

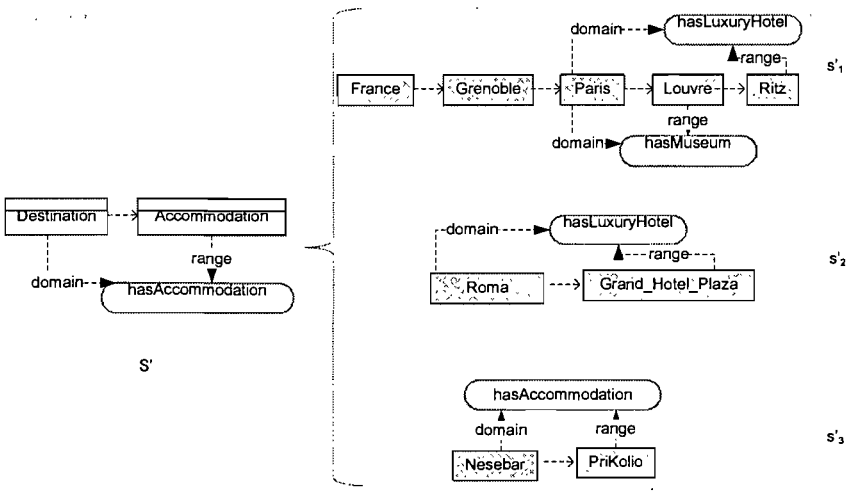


Figure 4.6 – Séquence de classes étendue et quelque-unes de ses instances.

$[S_2]_{\Gamma_\Omega} \subseteq [S_1]_{\Gamma_\Omega}$. *En d'autres mots, la relation de généralité se traduit par une inclusion des ensembles d'interprétation des motifs.*

La notion de généralité de motifs, telle que donnée ci-dessus, exige la disponibilité de toutes les séquences d'objets étendues couvertes par un motif. Dans les applications réalistes, seulement une partie de l'univers des séquences d'objets étendues est disponible. Ainsi, la relation de généralité est impraticable. Pour ces raisons, nous avons défini une relation qui s'inspire de la logique du premier ordre [15] et qui se base sur la syntaxe pour calculer la généralité entre motifs. Cette relation, appelée *subsomption*, est équivalente à la généralisation sémantique. La définition que nous utilisons est similaire à celle de l'instanciation qui se base sur la similarité des structures entre les motifs et les séquences d'objets étendues.

4.2.6 Subsomption de motifs

La subsomption de motifs est un ordre partiel qui nous permet d'ordonner un ensemble de motifs en hiérarchies suivant le niveau de généralité (voir la proposition 4.2.1).

Intuitivement, deux motifs sont liés par la relation de subsomption si l'un d'eux se manifeste dans l'autre de manière identique ou avec des concepts et des relations plus spécifiques. En d'autres mots, il s'agit de retrouver un prédécesseur du premier motif qui est identiquement présent dans le deuxième motif. La formalisation de cette relation est donnée par la définition 4.2.6 ci-dessous.

Remarque 4.2.4.

Pour des raisons de concision, l'expression $\exists\ c \in ran(r', S.\varsigma[i])$ *tel que* $S.\varsigma[j] \sqsubseteq_\Omega c$ *sera remplacée par* $(S.\varsigma[i], r', S.\varsigma[j]) \in \rho$.

Remarque 4.2.5.

Si $r(i,j)$ *est un triplet d'une séquence de classes étendue S et* r' *est une autre relation de* $\mathscr{R}_\Omega$ *alors, pour des raisons de concision, nous écrivons* $r(i,j) \sqsubseteq_\Omega r'(i,j)$ *lorsque :*

1. $r \sqsubseteq_\Omega r'$;
2. $(S.\varsigma[i], r', S.\varsigma[j]) \in \rho$.

Définition 4.2.6. ***Subsomption de motifs***

Soient S_1 *et* S_2 *deux motifs de* Γ_Ω*, on dit que* S_2 *subsume* S_1*, noté* $S_1 \sqsubseteq_{\Gamma_\Omega} S_2$*, s'il existe une fonction monotone injective croissante* $\psi : [1..|S_2.\varsigma|] \rightarrow [1..|S_1.\varsigma|]$ *tel que les deux conditions suivantes soient satisfaites :*

1. $\forall i \in [1..|S_2.\varsigma|],\ S_1.\varsigma[\psi(i)] \sqsubseteq_\Omega S_2.\varsigma[i]$,
2. $\forall r(i_1, i_2) \in S_2.\theta,\ \exists\ (j_1, j_2) \in ([1..|S_1.\varsigma|])^2 :\ \psi(i_1) = j_1,\ \psi(i_2) = j_2$ *et* $\exists\, r'(j_1, j_2) \in S_1.\theta$ *tel que* $r' \sqsubseteq_\Omega r$.

La fonction ψ, permet entre autres, d'associer à chaque position de $[1..|S_2.\varsigma|]$ une position unique dans l'ensemble d'arrivée $[1..|S_1.\varsigma|]$ correspondant à un concept plus spécifique. De plus, deux positions différentes dans S_2 ont deux images différentes dans

S_1, ce qui fait de ψ un fonction injective. Ainsi, des concepts différents (resp. des relations différentes) du motif S_2 sont associé(e)s avec des concepts (resp. relations) plus spécifiques différent(e)s dans le motif S_1. Afin que la mise en correspondance entre les concepts de deux séquences étendues soit co-linéaire, ψ est monotone croissante.

La figure 4.7 illustre la subsomption de motifs. Les lignes grises discontinues représentent la mise en correspondance entre les éléments des motifs.
On y voit que pour le motif S'', le concept *Accommodation* et la relation *hasAccommodation* n'ont pas d'antécédents dans le motif S. Ainsi, S'' ne subsume pas S. Par contre, dans le cas de S', tous les concepts et relations ont un antécédent dans S tel que spécifié dans la définition 4.2.6.

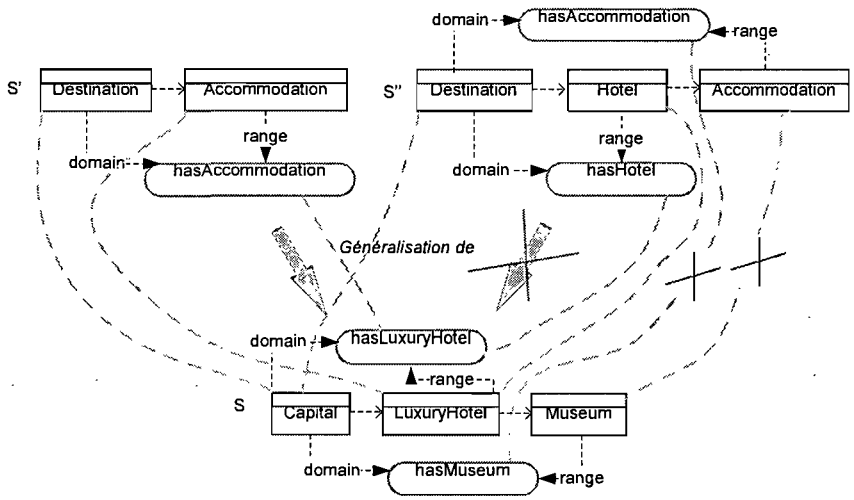


Figure 4.7 – Exemple de subsomption de motifs.

L'exemple de la figure 4.7 illustre la subsomption de motifs suivant l'ontologie *Travel*. Sur cette figure, nous constatons que pour les motifs S' et S, les conditions de la définition 4.2.6 sont satisfaites, et ainsi nous concluons que le motif S' subsume S. En effet, pour les concepts *Capital* et *LuxuryHotel* de S correspond respectivement les concepts les plus généraux de S' *Destination* et *Accommodation*. De même, la relation *hasCapital* de S est associée avec la relation *hasAccommodation* de S'. En revanche, S'' ne subsume

pas S à cause de l'existence d'une classe dans S'' (à savoir *Accommodation*) qui n'a pas d'antécédent dans S.

Remarque 4.2.6. *Pour tout motif S de Γ_Ω, nous avons $S \sqsubseteq_{\Gamma_\Omega} \emptyset_{\Gamma_\Omega}$, c'est-à-dire que le motif vide est la racine de l'espace $\sqsubseteq_{\Gamma_\Omega}$.*

Il est à noter que la relation de subsomption que nous avons définie est un ordre partiel. Cela signifie, entre autres, que $S_1 = S_2$ si et seulement si $S_1 \sqsubseteq_{\Gamma_\Omega} S_2$ et $S_2 \sqsubseteq_{\Gamma_\Omega} S_1$ (voir la proposition 4.2.1). De plus, cette relation (relation de subsomption) est une relation réflexive, transitive et anti-symétrique (voir les propositions 4.2.2, 4.2.3, 4.2.4).

Proposition 4.2.1. ***Ordre partiel***

La relation $\sqsubseteq_{\Gamma_\Omega}$ est un ordre partiel sur l'ensemble des motifs Γ_Ω.

Proposition 4.2.2. ***Réflexivité de la relation de subsomption***

La relation $\sqsubseteq_{\Gamma_\Omega}$ est une relation réflexive sur l'ensemble Γ_Ω. Autrement dit : $\forall S \in \Gamma_\Omega, S \sqsubseteq_{\Gamma_\Omega} S$.

Proposition 4.2.3. ***Transitivité de la relation de subsomption***

La relation de subsomption est une relation transitive. Autrement dit, soient S_1, S_2 et S_3 trois motifs de Γ_Ω, alors :

$$(S_1 \sqsubseteq_{\Gamma_\Omega} S_2 \wedge S_2 \sqsubseteq_{\Gamma_\Omega} S_3) \Rightarrow (S_1 \sqsubseteq_{\Gamma_\Omega} S_2)$$

Proposition 4.2.4. ***Anti-symétrie de la relation de subsomption***

La relation $\sqsubseteq_{\Gamma_\Omega}$ est une relation anti-symétrique. Autrement dit : $\forall S_1, S_2 \in \Gamma_\Omega^2, S_1 \sqsubseteq_{\Gamma_\Omega} S_2 \wedge S_2 \sqsubseteq_{\Gamma_\Omega} S_1 \Rightarrow S_1 = S_2$

Nous pouvons maintenant énoncer que la subsomption peut être utilisée pour tester la généralisation des séquences de classes étendues (voir la proposition 4.2.5).

Proposition 4.2.5. *Soient S_1 et S_2 deux séquences de classes étendues de Γ_Ω tel que $S_1 \sqsubseteq_{\Gamma_\Omega} S_2$, alors $S_1 \leq S_2$.*

Pour la démonstration des cinq propositions ci-haut mentionnées, le lecteur pourra se référer aux annexes I.2 à I.6.

Maintenant que la subsomption remplace effectivement la notion de généralité entre motifs, il n'est plus nécessaire de disposer des ensembles d'interprétation complets pour comparer deux motifs. À présent, la question qui se pose est de savoir comment vérifier que deux motifs sont liés par la relation se subsomption.

Dans la suite, nous présentons deux alternatives pour calculer la subsomption de deux motifs. La première alternative consiste à construire un algorithme qui teste la subsomption entre deux motifs suivant la définition 4.2.6. La deuxième solution consiste à définir un modèle de dérivation de la subsomption qui se base sur un ensemble de quatre opérations.

4.2.6.1 Test de subsomption

Nous présentons un algorithme qui teste si un motif est plus général qu'un autre motif en vérifiant que le premier motif subsume le deuxième motif. Cet algorithme, appelé SUBTEST (voir l'algorithme 6), est une adaptation de l'algorithme 3 qui teste l'instanciation d'une séquence de classes étendue par une séquence d'objets étendue. Principalement, il s'agit de remplacer la séquence d'objets étendue par une séquence de classes étendue et le test d'appartenance d'un objet (resp. d'un lien) à l'interprétation d'un concept (resp. d'une relation) par un test d'instanciation entre deux concepts (resp. entre deux relations). Ainsi, son principe se résume comme suit : pour chaque classe et pour chaque relation de la première séquence, l'algorithme recherche une image dans la

deuxième séquence et s'assure que les images des concepts et des relations préservent les contraintes structurelles.

Étant donné que pour un concept (resp. une relation) plusieurs images potentielles peuvent être retrouvées , l'algorithme considère chacune de ces images potentielles jusqu'à ce que une image appropriée soit retrouvée ou que tous les concepts/relations soient rejetées. Afin de réduire la complexité du test de subsomption, une optimisation similaire à celle apportée au test d'instanciation a été implantée. Il s'agit de traiter les correspondances des relations après avoir retrouvé les images potentielles des concepts source et destination de la relation.

4.2.7 Calcul de la subsomption

Nous définissons quatre opérations sur les motifs qui conjointement représentent un système complet et valide de calcul de la subsomption. Chaque opération consiste en une transformation simple qui permet de passer d'un premier motif à un deuxième motif. En d'autres mots, deux motifs sont liés par la relation de généralisation si et seulement si un de ces motifs peut être obtenu à partir de l'autre motif en appliquant un nombre fini de fois les opérations de l'ensemble.

Nous définissons les quatre opérations en trois étapes où à chaque étape on raffine les définitions des opérations de l'étape qui la précède. Nous commençons par la définition du premier ensemble des opérations qu'on appelle *opérations génériques* (section 4.2.7.1). Ces opérations admettent la spécialisation d'un concept (resp. relation) par un concept (resp. relation) qui n'est pas systématiquement un de ses successeurs immédiat dans l'ontologie. D'autres opérations plus précises sont présentées dans le chapitre 5. Ainsi, dans la section 5.1.2 de ce chapitre, nous présentons l'ensemble des *opérations élémentaires* où seulement la spécialisation par des successeurs immédiats est permise. Un troisième ensemble d'opérations, dites *opérations canoniques*, sont présentées dans la section 5.1.3 du même chapitre. Ces opérations sont le résultat de l'application des opérations élémentaires à un endroit précis d'un motif.

Algorithm 6 Procédure de test de subsomption

```
1: procedure SUBTEST(S2, S1 : deux séquences de classes étendue)   ▷ On cherche à savoir si S1 ⊑Ω S2
2:     if |S2.ς| > |S1.ς| ou |S2.θ| > |S1.θ| then
3:         return false;
4:     end if
5:     imgsPot ;          ▷ Tableau de listes contenant les images potentielles de chaque concept
6:     imgsPot ← TROUVERIMAGESPOTENTIELLES(S2,S1);
7:     i ← 1 ;
8:     while i ≤ |S2.ς| do
9:         if imgsPot[i].next? == false then
10:            if i == 1 then
11:                return false;          ▷ On n'a pas pu apparier le premier concept
12:            else
13:                imgsPot[i].toFirst ; ▷ Pointer vers la première position de imgsPot[i]
14:                i − − ;
15:                break ;
16:            end if
17:        end if
18:        posImagCrt ← imgsPot[i].next() ;
19:        if i > 1 et posImagCrt < imgsPot[i − 1].current() then
20:            break ;
21:        end if
22:        if RELATIONSCORRESPONDENT(S2,S1,i,imgsPot,posImagCrt) then
23:            i + + ;
24:        end if
25:    end while
26:    return true;
27: end procedure
```

Algorithm 7 Procédure qui vérifie s'il y a correspondance de relations de deux motifs

1: **procedure** RELATIONSCORRESPONDENT(S_2, S_1 : deux motifs, i : position d'un concept dans S_2, *imgsPot* : images potentielles, *posImagCrt* : position de l'image du concept de la position i)
2: **for all** $r(k,i)$ **in** $S_2.\theta$ **do**
3: **if** $\nexists r'(imgsPot[k].current(), posImagCrt) \in S_1.\theta$ et $r' \sqsubseteq_\Omega r$ **then**
4: **return** *false*;
5: **end if**
6: **end for**
7: **return** *true*;
8: **end procedure**

Algorithm 8 Procédure de recherche d'images potentielles entre deux motifs

1: **procedure** TROUVERIMAGESPOTENTIELLES(S_2, S_1 : deux motifs)
2: **for** $i = 1 ; i \leq |S_1.\varsigma| ; i++$ **do**
3: **for** $j = 1 ; j \leq |S_2.\varsigma| ; j++$ **do**
4: **if** $(S_1.\varsigma[i] \sqsubseteq_\Omega S_2.\varsigma[j])$ et $\forall r(i,k) \in S_2.\theta, \exists r'(j,k') \in S_1.\theta r(i,k) \sqsubseteq_\Omega r'(j,k')$ et $\forall r(k,i) \in S_2.\theta, \exists r'(k',j) \in S_1.\theta\ r(k,i) \sqsubseteq_\Omega r'(k',j)$ et $(|S_1.\varsigma| - j \geq |S_2.\varsigma| - i)$ **then**
5: $imgsPotTemp[j].addElement(i)$;
6: **end if**
7: **end for**
8: **end for**
9: **return** *imgsPotTemp*;
10: **end procedure**

4.2.7.1 Opérations générales

La première opération, dénotée par *splCls*(), consiste à spécialiser une classe, c'est-à-dire, remplacer une classe se trouvant à une position donnée dans une séquence par une classe plus spécifique de l'ontologie du domaine. La deuxième opération, dénotée par *addCls*(), insère une nouvelle classe à une position donnée d'une séquence. La troisième opération, dénotée par *splRel*(), consiste à spécialiser une relation individuelle, c'est-à-dire, remplacer une relation à un endroit donné par une relation plus spécifique de l'ontologie. Dans la quatrième et dernière opération, dénotée par *addRel*(), il s'agit d'ajouter une relation à deux positions correspondant à des classes de la séquence.

Étant donné que les motifs sont une représentation abstraite des séquences d'objets étendues, et que dans ces dernières, les objets contigus peuvent être identiques ou comparables, on autorise à ce que les concepts contigus d'un motif soient comparables ou identiques. De même que pour les séquences d'objets étendues, où entre deux objets on ne garde que la relation la plus spécifique, les relations entre deux concepts d'un motif ne peuvent être identiques ou comparables.

La spécialisation d'une classe s'effectue par le remplacement d'une classe c à une position j d'une séquence $S = (\varsigma, \theta)$ avec une classe c' de $\mathscr{C}_\Omega$ tels que c' est plus spécifique que c dans l'ontologie Ω, $c' \sqsubseteq_\Omega c$, c' est comparable avec les classes co-domaines de toutes les relations $r(*, j)$ de θ où le symbole * représente une valeur quelconque de l'intervalle de valeurs possibles. Formellement, nous avons ce qui suit :

Définition 4.2.7. ***Spécialisation de classe***

Soient $S \in \Gamma_\Omega$, $c \in \mathscr{C}_\Omega$ *et* j *un entier tel que* $j \leq |S.\varsigma|$ *et où* $c \sqsubseteq_\Omega S.\varsigma[j]$. *Alors, l'opération* $splCls(S, c, j)$ *donne lieu à la séquence* (ς, θ) *où :*

- $\varsigma = \langle S.\varsigma[1], ..., S.\varsigma[j-1], c, S.\varsigma[j+1], ..., S.\varsigma[|S.\varsigma|]\rangle$;
- $\theta = S.\theta$,

Par exemple, en replaçant la classe *Activity* par la classe la plus spécifique *Sightseeing* dans la séquence $(\langle Destination, Activity\rangle, \{hasActivity(1,2)\})$, nous obtenons la séquence $(\langle Destination, Sightseeing\rangle, \{hasActivity(1,2)\})$.

L'insertion de la classe c de $\mathscr{C}_\Omega$ dans la séquence S à la position j est possible si j est dans la limite de la taille de la séquence: De manière formelle, l'opération d'ajout de classe est donnée par la définition 4.2.8 ci-dessous.

Définition 4.2.8. ***Ajout de classe***

Soient $S \in \Gamma_\Omega$, $c \in \mathscr{C}_\Omega$ *et* j *un entier tel que* $1 \leq j \leq |S.\varsigma| + 1$. *Alors, l'opération* $addCls(S,c,j)$ *conduit à la génération de la séquence* (ς, θ) *où :*

- $\varsigma = \langle S.\varsigma[1], ..., S.\varsigma[j-1], c, S.\varsigma[j], ..., S.\varsigma[|S.\varsigma|]\rangle$;
-

$$\theta = \begin{cases} \textit{// relations dont les positions ne cheangent pas} \\ \{r(l,m) | r(l,m) \in S.\theta, 1 \leq l \leq m < j\} \\ \cup \\ \textit{//relations dont seulement la dernière position est décalée} \\ \{r(l,m+1) | r(l,m) \in S.\theta, 1 \leq l < j \leq m\} \\ \cup \\ \textit{//relations dont les deux positions sont décalées.} \\ \{r(l+1,m+1) | r(l,m) \in S.\theta, 1 \leq j \leq l \leq m\} \end{cases}$$

À titre d'exemple, nous avons : $addCls((\langle Destination, Activity\rangle, \{hasActivity(1,2)\}), Hotel, 2) = (\langle Destination, Hotel, Activity\rangle, \{hasActivity(1,3)\})$.

L'ajout d'un lien relationnel r de $\mathscr{R}_\Omega$, entre les deux classes se situant respectivement aux positions i et j, d'une séquence S est possible si ces positions sont proprement ordonnées et qu'elles sont contenues dans $[1..|S.\varsigma|]$, et que les classes correspondantes sont au moins des sous-classes d'une paire valide de classes domaine et co-domaine de r dans l'ontologie Ω. De plus, il est exigé à ce qu'il n'existe aucune relation r' dans la

séquence avec les mêmes indices tel que r' est une spécialisation de r.

Définition 4.2.9. ***Ajout de relation***

Soient $S \in \Gamma_\Omega$, $r \in \mathscr{R}_\Omega$ *et* i, j *deux entiers tel que :*

1. $1 \leq i \leq j \leq |S.\varsigma|$;
2. $S.\varsigma[i] \ll dom(r)$;
3. $S.\varsigma[j] \ll ran(r, S.\varsigma[i])$;
4. $\nexists\, r'(i,j) \in S.\theta$ *tel que* $r'(i,j) \sqsubseteq_\Omega r(i,j)$.

Alors, l'opération $addRel(S, r, i, j)$ *donne lieu à la séquence* (ς, θ) *où :*

- $\varsigma = S.\varsigma$;
- $\theta = S.\theta \cup \{r(i,j)\}$.

L'exemple suivant, illustre l'insertion de la relation *hasMuseum* dans le motif donné : $addRel((\langle Destination, Museum\rangle, \{\}), hasMuseum, 1, 2) = (\langle Destination, Museum\rangle, \{hasMuseum(1,2)\})$.

La spécialisation d'une relation consiste à remplacer $r(i,j)$ d'une séquence $S = (\varsigma, \theta)$ avec une autre relation $r'(i,j)$ tel que le domaine d'interprétation de $r'(i,j)$ est un sous-ensemble de l'ensemble d'interprétation de $r(i,j)$. Cette définition est formalisée comme suit :

Définition 4.2.10. ***Spécialisation de relation***

Soient $S \in \Gamma_\Omega$, $r \in \mathscr{R}_\Omega$ *et* i, j *deux entiers tel que* $r(i,j) \in S.\theta$ *et* $r'(i,j) \sqsubseteq_\Omega r(i,j)$. *Alors, l'opération* $splRel(S, r, i, j, r')$ *donne lieu à la séquence* ς, θ *où :*

- $\varsigma = S.\varsigma$;
- $\theta = (S.\theta - \{r(i,j)\}) \cup \{r'(i,j)\}$.

À titre illustratif, on a : $splRel((\langle Capital, Museum\rangle, \{hasActivity(1,2)\}), hasActivity, 1, 2, hasMuseum) = (\langle Capital, Museum\rangle, \{hasMuseum(1,2)\})$.

La proposition 4.2.6 stipule que les opérations générales sont suffisantes pour calculer la subsomption.

Proposition 4.2.6. ***Complétude de l'ensemble des opérations de spécialisation***

Considérons S_1 et S_2 deux séquences de Γ_Ω, alors $S_1 \sqsubseteq_{\Gamma_\Omega} S_2$ si et seulement si, la séquence S_1 peut être obtenue en appliquant à S_2 un nombre fini de fois les opérations de spécialisation de motifs (ajout de concept, ajout de relation, spécialisation de concept et spécialisation de relation).

La preuve de cette proposition est présentée dans l'annexe I.1.

La figure 4.8 présente un exemple qui illustre l'utilisation des opérations générales pour calculer la subsomption entre les motifs S_1 et S_2.

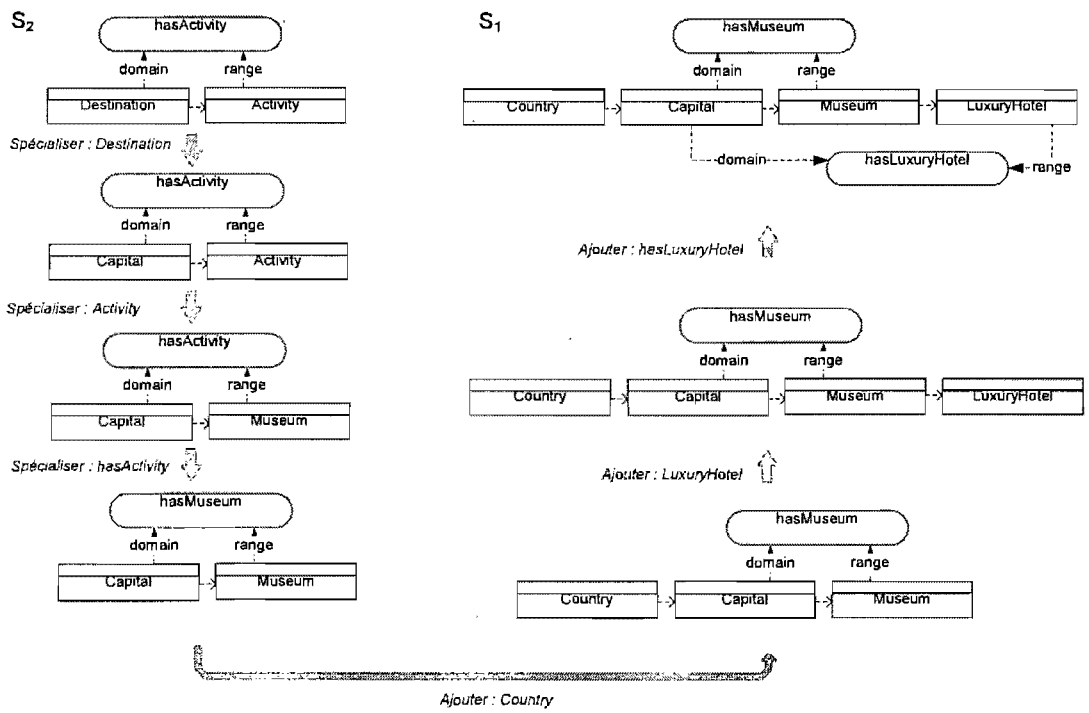


Figure 4.8 – Calcul de la subsomption par les opérations générales.

4.3 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle catégorie de motifs pour la fouille de données ontologiques. En effet, ces motifs intègrent les connaissances ontologiques. Ainsi, nous avons présenté deux langages, le premier pour la représentation des données et le deuxième pour la représentation des motifs. C'est ainsi que nous avons séparé les aspects purement structuraux des motifs des aspects sémantiques. Par la suite, nous avons présenté les relations existantes entre les motifs et les séquences d'objets étendues (relation d'instanciation et relation d'interprétation) ainsi que la relation de subsomption qui permet d'organiser les motifs en hiérarchies suivant leur niveau de généralité. Afin de se déplacer dans ces hiérarchies, nous avons construit un ensemble d'opérations de spécialisation de motifs. Ces opérations prises ensemble constituent une solution complète pour le calcul de la subsomption. Pour compléter notre méthodologie, nous avons appuyé les différentes affirmations par un ensemble de propositions dont la preuve se trouve en annexe.

Chapitre 5

Solutions algorithmiques de la fouille de motifs séquentiels

Dans ce chapitre, nous présentons le développement d'une solution algorithmique pour la fouille de la nouvelle catégorie de motifs présentée dans le chapitre 4. La méthode de fouille proposée s'inspire de l'algorithme *Apriori* [75] qui parcourt l'espace des motifs par niveau, en allant des motifs les plus généraux aux motifs les plus spécifiques en se basant sur une relation de généralisation qui induit une structure hiérarchique partiellement ordonnée sur l'ensemble de tous les motifs.

5.1 Fouille des motifs séquentiels

Soient une ontologie Ω, l'univers des objets (items) de cette ontologie $\mathscr{O}_\Omega$, et une base de données $\mathscr{D}$, composée d'enregistrements qui combinent les items de $\mathscr{O}_\Omega$ ($\mathscr{D} \subseteq O^\omega$), et $\mathscr{D}_\Omega$ la base des séquences d'objets étendues associée. L'espace Γ_Ω est composé de toutes les séquences de classes étendues construites par-dessus l'ontologie Ω. L'extraction des motifs fréquents consiste à rechercher les motifs de l'espace Γ_Ω dont la fréquence de manifestation au sein de la base $\mathscr{D}_\Omega$ est supérieure ou égale à un seuil fixe σ, appelé aussi *support minimal*. Il est à signaler que plusieurs critères peuvent être pris en compte dans le calcul de ce support. À titre d'exemple, un poids représentant le nombre de fois qu'un motif se manifeste dans une séquence d'objets, peut être intégré dans la mesure du support. Dans notre cas, le support se calcule en considérant les deux cas suivants : soit que le motif se manifeste dans la séquence d'objets et dans ce cas le support est augmenté de 1, soit que le motif n'apparaît pas dans la séquence, auquel cas la valeur du support reste inchangée.

La famille des motifs de $\mathscr{D}_\Omega$ ayant un support supérieur ou égal à σ est représentée par $\mathscr{F}_\sigma$. Pour calculer la fréquence d'un motif, l'algorithme 3 du chapitre 4 est utilisé.

Une fouille naïve des motifs de $\mathscr{F}_\sigma$ consiste à tester toutes les combinaisons possibles des motifs afin d'extraire ceux qui sont suffisamment fréquents. Étant donné que nous tenons compte des hiérarchies de concepts et des hiérarchies de relations, le nombre de motifs à tester peut être plusieurs fois supérieur au nombre de motifs à tester si seulement les objets sont considérés (voir l'annexe II). De plus, considérer toutes les combinaisons possibles sous-entend la génération de motifs non valides, c'est-à-dire, des séquences de concepts avec des relations qui n'impliquent pas nécessairement les concepts aux positions indiquées.

Pour retrouver efficacement les motifs fréquents, une stratégie de parcours de l'espace des motifs devra être adoptée. En effet, un parcours astucieux de l'espace des motifs permettra de considérer seulement une partie de l'espace et du coup réduire la surcharge en calcul et ainsi réduire le temps nécessaire pour retrouver tous les motifs fréquents. Ainsi, nous optons pour un parcours par niveau de l'espace de motifs, où les motifs non valides ne sont pas testés.

5.1.1 Partitionnement de l'espace des motifs

Pour assurer un parcours homogène de l'espace des motifs, on se propose de subdiviser cet espace en niveaux où chaque niveau ne contient que les motifs de même degré de généralité. Ce dernier se base sur une mesure cumulative des différents degrés de généralité des éléments composant un motif (concepts et relations).

Fouille de motifs par niveaux

Pour motiver le partitionnement en niveaux de l'espace des motifs, considérons le treillis booléen de tous les sous-ensembles d'items (itemsets). Les itemsets sont séparés en niveaux suivant leur taille, et les successeurs immédiats d'un motif p par rapport à la relation de généralité $\sqsubseteq_{\Gamma_\Omega}$ sont tous les super-itemsets p' qui ont exactement un item de

plus que p. Ainsi, ils tombent dans le niveau $k+1$ du treillis, avec p un motif du niveau k (ayant k items).

Apriori [10] est l'algorithme de recherche de motifs fréquents de référence qui effectue un parcours par niveau du haut en bas du treillis booléan des itemsets.

En effet, l'espace des motifs est hiérarchiquement organisé par rapport à la relation de généralité définie sur les motifs de manière comparable à la relation d'inclusion ensembliste. Dans cette hiérarchie, les niveaux se distinguent par la cardinalité des motifs qu'ils contiennent. Cela signifie qu'au niveau k on ne retrouve que les motifs de taille k. Dans *Apriori*, un seul niveau est traité à la fois. Afin de ne pas tester tous les motifs de l'espace, cet algorithme exploite la propriété d'anti-monotonie de la fréquence (si un motif du niveau k n'est pas fréquent alors tout motif du niveau k+1 qui est une spécialisation du motif courant n'est pas fréquent). Ainsi, l'exploration d'une branche de l'espace est arrêtée lorsqu'un motif non-fréquent est retrouvé. Par conséquent, on évite de parcourir l'espace des itemsets en entier et ainsi on réduit le nombre de fois que la base de départ sera parcourue. De plus, cette exploration par niveau garantit qu'aucun motif fréquent n'est oublié, car les sous-ensembles d'un motif fréquent sont tous fréquents et de ce fait il n'y aura pas d'élagage de ce motif, ou de ses sous-ensembles.

La recherche de motifs fréquents par niveau à la *Apriori* a été adaptée avec succès pour d'autres types de structures telles que les séquences [11], les arbres [89], les graphes [44], ainsi que pour la fouille des hiérarchies de concepts bâties autour d'items individuels (fouille de motifs généralisés) [76].

Le dénominateur commun à toutes ces approches réside dans l'extension des éléments propres à la fouille des itemsets à des environnements plus riches. Ainsi, de la structure de treillis on passe à la relation d'ordre partiel, et de la taille des motifs à une mesure de la généralité (rang).

Une fois que la notion de niveau est définie, la recherche de la famille de motifs $\mathscr{F}_\sigma$ à travers l'espace Γ_Ω muni de la relation $\sqsubseteq_{\Gamma_\Omega}$, qu'on notera par $\langle \Gamma_\Omega, \sqsubseteq_{\Gamma_\Omega} \rangle$, peut s'effectuer en explorant la monotonie de la fréquence des motifs par rapport à la relation de généralité $\sqsubseteq_{\Gamma_\Omega}$: un super-motif d'un motif fréquent est fréquent.
Il est à noter que tous les exemples de ce chapitre se basent sur l'ontologie *Travel* dont

une vue partielle est représentée par la figure 3.3 du chapitre 3.

Approche de fouille par niveaux adaptée aux données décrites par une ontologie

Pour plusieurs raisons, l'algorithme *Apriori* ne peut être directement appliqué à la fouille des motifs composés de concepts et de relations d'une ontologie du domaine. L'une de ces raisons, est qu'*Apriori* utilise la taille d'un itemset pour représenter le degré de généralité de ce dernier, et par la même, du niveau auquel il appartient. Or, dans notre cas, le degré de généralité d'un motif n'est pas seulement déterminé par le nombre de concepts/relations qui le composent, mais aussi par leur degré de généralité. En effet, le niveau ne dépend pas seulement de la taille du motif (nombre de concepts et de relations) du moment que deux motifs de Γ_Ω ayant la même taille peuvent être liés par une relation de généralité. Par exemple, les deux motifs $(\langle Destination, Accommodation\rangle, \{hasAccommodation(1,2)\})$ et $(\langle City, Hotel\rangle, \{hasHotel(1,2)\})$ ont le même nombre de concepts mais le premier est de niveau d'abstraction plus élevé que le deuxième, car *Destination* est une généralisation de *City*, *Accommodation* est la généralisation de *Hotel*, et $hasAccommodation(1,2)$ est la généralisation de $hasHotel(1,2)$.

Afin de pouvoir exploiter le principe d'*Apriori* pour la fouille des motifs ontologiques (motifs de concepts/relations), ses éléments de base devraient être adaptés. En premier lieu, une fonction monotone basée sur la relation de généralité entre motifs est à développer. Afin de pouvoir l'exploiter efficacement, cette fonction de calcul du rang, devrait permettre d'attribuer à chaque motif d'un niveau la même valeur. En outre, un motif est, selon le degré de généralité, le prédécesseur immédiat d'un autre motif, et donc son rang est égal au rang de son successeur moins 1.

Cependant, et à notre connaissance, de toutes les extensions d'*Apriori* qui existent, aucune ne propose une fonction pour le calcul du rang qui couvre la fouille de motifs ontologiques. Pour couvrir cette catégorie de motifs, une fonction de rang devrait prendre en considération les deux facteurs suivants. Premièrement, la généralité d'un élément individuel (concept et relation) devrait être considérée. Deuxièmement, la généralité d'un

ensemble d'éléments (concepts, relations) devrait entrer dans le calcul du rang. Alors que la généralité d'un élément reflétera la position de ce dernier dans la hiérarchie correspondante dans l'ontologie, la généralité d'un ensemble d'éléments permettra de cumuler les degrés de généralité des éléments qui le composent.

Cependant, il n'existe pas de définition standard du degré de généralité d'un concept ou d'une relation. Du point de vue de la théorie des graphes, on pourrait bien choisir entre le chemin le plus court ou le plus long entre un noeud cible et un élément de référence. Comme nous avons besoin de calculer une généralité relative, nous sommes intéressés d'avoir une mesure monotone, de telle sorte qu'un motif plus spécifique aura une profondeur plus grande. Ainsi, nous choisissons le plus long chemin, car le plus court chemin n'est pas nécessairement monotone. En effet, il est possible d'avoir plusieurs chemins de profondeurs différentes pour un même concept (cas de l'héritage multiple). Concrètement, nous associons à un concept d'une ontologie un indice de généralité qui correspond à la longueur du plus long chemin qui sépare ce concept de l'élément *Thing* [1]. De même, nous associons à chaque relation un indice de généralité qui correspond à la longueur du plus long chemin, qui la sépare de la relation racine dans la taxonomie correspondante, augmenté de 1. Il suffit maintenant de cumuler les indices de généralité des éléments qui composent un motif pour obtenir son rang. La définition mathématique de ce rang commence par la définition des mesures ci-dessous sur les concepts et les relations. Ces mesures permettent de calculer les profondeurs des éléments associés. Formellement, ces deux mesures sont définies comme suit.

Définition 5.1.1. ***Profondeur d'un élément d'un motif (concept, relation)***

Soient c un concept de $\mathscr{C}_\Omega$, et r une relation de $\mathscr{R}_\Omega$. Les fonctions de profondeur pour les concepts et les relations, P_c et P_r, sont respectivement définies dans Ω comme suit :

– $P_c : \mathscr{C}_\Omega \rightarrow \mathbb{N}$

$$P_c(c) = \begin{cases} 0, & si\ pred_\Omega(c) = \varnothing \\ max(\{P_c(c') | c \sqsubset_\Omega c'\}) + 1, & sinon. \end{cases}$$

1. *Thing* est le concept racine d'une ontologie.

- $P_r : \mathscr{R}_\Omega \rightarrow \mathbb{N}$

$$P_r(r) = \begin{cases} 1, & si\ pred_\Omega(r) = \varnothing \\ max(\{P_r(r') | r \sqsubseteq_\Omega r'\}) + 1, & sinon. \end{cases}$$

Étant donné que la relation de généralisation entre concepts (resp. entre relations) est acyclique, alors les profondeurs définies ci-dessus donnent toujours des valeurs finies.

Il est à constater que la profondeur d'un concept (resp. d'une relation) est basée sur la valeur maximale de ses prédécesseurs. Cette définition est justifiée par le fait qu'un concept ou une relation peuvent avoir des prédécesseurs qui ont différents niveaux d'abstraction. Dans une telle situation, il sera inconsistant de considérer arbitrairement l'un d'eux et d'ignorer les autres. En prenant la valeur maximale de tous les prédécesseurs, le problème est résolu. Par exemple, dans l'ontologie *Travel*, nous avons le concept *Safari* qui est une spécialisation des concepts *Destination*, *Adventure*, et de *Sightseeing* puisque *Destination* est au niveau *1* alors que les concepts *Adventure* et *Sightseeing* sont au niveau *2*. Ainsi, la profondeur du concept *Safari* sera *3* ($max(2,2,1)+1$). Par conséquent, la définition ci-dessus assure que chaque élément a une valeur de profondeur qui est au moins une unité plus grande que la valeur maximale des profondeurs de ces prédécesseurs.

Remarque 5.1.1. *Dans le cadre de cette thèse, nous appelons élément racine un concept (resp. une relation) ayant une profondeur égale à 1.*

En se basant sur la notion de profondeur d'un concept/relation, nous introduisons les notions de prédécesseurs/successeurs réguliers d'un concept/relation.
Un concept est dit *prédécesseur* (resp. *successeur*) *régulier* d'un deuxième concept s'il est plus général (resp. plus spécifique) que ce dernier et la différence de leur profondeur est de 1. Une définition plus formelle est fournie ci-après

Définition 5.1.2. ***Prédécesseur régulier d'un concept***

Soient c, c' deux concepts de $\mathscr{C}_\Omega$, c' est dit prédécesseur régulier de c, noté $c \sqsubseteq^r_\Omega c'$, si et seulement si :

- $c \sqsubseteq_\Omega c'$;
- $P_c(c) = P_c(c') + 1$.

Définition 5.1.3. ***Successeur régulier d'un concept***

Soient c, c' deux concepts de $\mathscr{C}_\Omega$, c' est dit successeur régulier de c, noté $c' \sqsubseteq^r_\Omega c$, si et seulement si :

- $c' \sqsubseteq_\Omega c$;
- $P_c(c') = P_c(c) + 1$.

De manière similaire, une relation est dite *prédécesseur* (resp. *successeur*) *régulier* d'une deuxième relation s'elle est plus générale (resp. plus spécifique) que cette dernière et la différence de leurs profondeur est de 1 (voir les définition 5.1.4 et 5.1.5)..

Définition 5.1.4. ***Prédécesseur régulier d'une relation***

Soient r, r' deux relations de $\mathscr{R}_\Omega$, r' est dite prédécesseur régulier de r, noté $r \sqsubseteq^r_\Omega r'$, si et seulement si :

- $r \sqsubseteq_\Omega r'$;
- $P_r(r) = P_r(r') + 1$.

Définition 5.1.5. ***Successeur régulier d'une relation***

Soient r, r' deux relations de $\mathscr{R}_\Omega$, r' est dite successeur régulier de r, noté $r' \sqsubseteq^r_\Omega r$, si et seulement si :

- $r' \sqsubseteq_\Omega r$;
- $P_r(r') = P_r(r) + 1$.

Nous pouvons maintenant définir la fonction du *rang* qui est derrière la structuration par niveaux de l'espace des motifs $\langle \Gamma_\Omega, \sqsubseteq_{\Gamma_\Omega} \rangle$. Le valeur du rang reflète le degré de généralité d'une séquence par l'accumulation des degrés de généralité des classes et des relations de cette séquence. Comme la combinaison est additive, elle reflète aussi, et de manière indirecte, le nombre de classes et de relations dans la séquence.

Techniquement parlant, le rang d'une séquence $\tau : \Gamma_\Omega \rightarrow \mathbb{N}$, est la somme des pro-

fondeurs de ses classes et de ses relations :

$$\tau(S) = \sum_{c \in S.\varsigma} P_c(c) + \sum_{r \in S.\theta} P_r(r).$$

Par exemple, et suivant l'ontologie utilisée dans la présente thèse, le rang du motif $S = (\langle Destination, Hotel \rangle, \{hasHotel(1,2)\})$ est : $\tau(S) = 5$, car *Destination* a une profondeur de 1, alors que *Hotel* et *hasHotel* ont tous les deux une profondeur de 2 dans leur hiérarchies respectives.

Le rang, comme on vient de le définir, est monotone décroissant avec la relation de généralisation. Cette constatation est renforcée par la proposition 5.1.1 donnée ci-dessous.

Proposition 5.1.1. ***La monotonie du rang***

Soient S_1, S_2 deux séquences de Γ_Ω. $S_1 \sqsubseteq_{\Gamma_\Omega} S_2$ implique que $\tau(S_1) \geq \tau(S_2)$.

La démonstration de la proposition ci-dessus est présentée dans l'annexe I.7.

Remarque 5.1.2. *Le rang de $\emptyset_{\Gamma_\Omega}$ est égal à zéro ($\tau(\emptyset_{\Gamma_\Omega}) = 0$).*

Maintenant, il nous est possible d'énoncer la définition formelle d'un niveau dans l'espace des motifs comme suit.

Définition 5.1.6. ***Partitionnement de l'espace des motifs en niveaux***

Soient Ω une ontologie, $\langle \Gamma_\Omega, \sqsubseteq_{\Gamma_\Omega} \rangle$ l'espace de motifs associé, et k un entier supérieur ou égal à 0. Les motifs d'un niveau k dans $\langle \Gamma_\Omega, \sqsubseteq_{\Gamma_\Omega} \rangle$, dénoté par $\mathcal{N}_k$, est un sous-ensemble de Γ_Ω tel que :

$$\mathcal{N}_k = \{S | S \in \Gamma_\Omega \wedge \tau(S) = k\}$$

Si nous retournons aux opérations générales sur les motifs, définies dans la section 4.2.7.1 du chapitre 4, nous constatons que le rang peut être augmenté par plus d'une unité

à la fois. Par conséquence, un motif de niveau k peut être généré à partir de plusieurs motifs différents des niveaux inférieurs par l'application des opérations générales. La figure 5.1 présente un exemple d'une génération multiple d'un motif à partir de motifs appartenant à différents niveaux d'abstraction. Les deux motifs de la partie haute de l'image se spécialisent pour donner lieu au motif du bas. Le motif en haut à gauche, de rang 4, donne lieu au motif de rang 6 $S = (\langle Destination, Safari \rangle, \{hasAdventure(1,2)\})$ en ajoutant la relation *hasAdventure* aux concepts *Destination* et *Safari*. Le deuxième motif, quant à lui, est de rang 5 et il permet de générer S en spécialisant le concept *Adventure* en *Safari*.

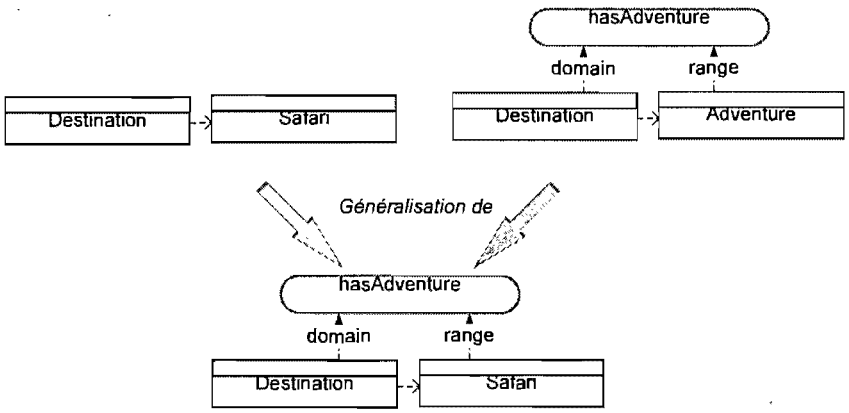


Figure 5.1 – Génération redondante à partir de motifs de niveaux différents.

Afin de remédier à ce problème, des opérations qui assurent un parcours niveau par niveau de l'espace des motifs sont proposées. Avec ces opérations, le rang n'est augmenté que d'une seule unité à la fois, ce qui signifie qu'en appliquant une de ces nouvelles opérations, appelées *opérations élémentaires*, le motif résultant se retrouvera exactement dans le niveau suivant celui du motif sur lequel est appliquée l'opération.

5.1.2 Opérations élémentaires

Les opérations élémentaires sont des versions des opérations générales déjà définies auxquelles nous avons ajouté des contraintes afin de n'augmenter le rang d'un motif que

d'une seule unité à la fois. Ces opérations sont dénotées par les noms des opérations générales suivis du suffixe *"E"*, et ainsi nous obtenons les quatre opérations suivantes : $addClsE()$, $addRelE()$, $splClsE()$ and $splRelE()$.

Tout d'abord, on autorise seulement l'ajout des classes les plus générales d'une hiérarchie de classes qui satisfont les conditions de la définition 4.2.8. Ceci donne lieu à la définition 5.1.7 suivante :

Définition 5.1.7. ***Ajout élémentaire de concept***

Soient $S \in \Gamma_\Omega$, $c \in \mathscr{C}_\Omega$ et j un entier. L'opération $addClsE(S,c,j)$ permet d'ajouter le concept c à la position j du motif S. Cette opération est définie par $addCls(S,c,j)$ seulement si $P_c(c) = 1$. Sinon elle est indéfinie.

Par exemple, $addClsE((\langle Destination, Activity\rangle, \{hasActivity(1,2)\}), Accommodation, 2) = (\langle Destination, Accommodation, Activity\rangle, \{hasActivity(1,3)\})$.

Pour la spécialisation élémentaire d'un concept, seulement le remplacement par un successeur immédiat dans l'ontologie est permis. De plus, pour respecter notre contrainte de niveau, la profondeur de ce concept doit avoir un rang supérieur d'une seule unité par rapport au rang du concept d'origine.

Définition 5.1.8. ***Spécialisation élémentaire de concept***

Soient $S \in \Gamma_\Omega$, $c \in \mathscr{C}_\Omega$ et j une position dans S. L'opération $splClsE(S,c,j)$ permet de spécialiser le concept qui est à la position j dans S par c. Cette opération est définie par $splCls(S,c,j)$ seulement si $c \sqsubseteq^r_\Omega S.\varsigma[j]$. Sinon, elle est indéfinie.

Par exemple, $splClsE((\langle Destination, Activity\rangle, \{\}), Sightseeing, 2) = (\langle Destination, Sightseeing\rangle, \{\})$.

De même, pour l'ajout de relation, on autorise seulement les relations générales qui satisfont les conditions de la définition 4.2.9.

Définition 5.1.9. ***Ajout élémentaire de relation***

Soient $S \in \Gamma_\Omega$, $r \in \mathscr{R}_\Omega$, i et j deux positions dans S. L'opération $addRelE(S,r,i,j)$ permet d'ajouter la relation r à S aux positions i et j. Cette opération est définie par $addRel(S,r,i,j)$ seulement si $P_r(r) = 1$. Sinon elle est indéfinie.

À titre d'exemple, nous avons : $addRelE((\langle Capital, LuxuryHotel\rangle, \{\}),$ $hasAccommodation, 1, 2) = (\langle Capital, LuxuryHotel\rangle, \{hasAccommodation(1,2)\})$.

Pour la spécialisation de la relation r, seulement un remplacement avec un successeur immédiat dans la hiérarchie à laquelle appartient r est autorisé. De plus, la profondeur d'une relation remplaçante doit être supérieure d'une unité de la profondeur de la relation remplacée.

Définition 5.1.10. ***Spécialisation élémentaire de relation***

Soient $S \in \Gamma_\Omega$ et $r, r' \in (\mathscr{R}_\Omega)^2$ et i, j deux positions dans S. L'opération $splRelE(S,r,i,j,r')$ permet de spécialiser la relation r ayant les positions i et j par r'. Cette opération est définie par $splRel(S,r,i,j,r')$ seulement si $r' \sqsubseteq^r_\Omega r$. Sinon, elle est indéfinie.

Par exemple, nous avons : $splRelE((\langle Capital, Museum\rangle, \{hasActivity\ (1,2)\}),$ $hasActivity, 1, 2, hasSightseeing) = (\langle Capital, Museum\rangle, \{hasSightseeing(1,2)\})$.

Malgré la restriction des opérations applicables aux opérations élémentaires, il peut encore subsister beaucoup de redondances quant à la génération d'un motif donné. En effet, par l'application des opérations élémentaires, plusieurs motifs de niveau k peuvent

donner lieu au même motif de niveau $k+1$. La figure 5.2 présente la génération multiple du motif $S = (\langle RuralArea, Safari \rangle, \{hasAdventure(1,2)\})$ par l'application d'opérations élémentaires différentes. La spécialisation des motifs S_1, S_2 et S_3 de la partie haute de l'image, qui sont de niveau 6, donne lieu au motif S du bas de l'image, qui est de rang 7. Le motif S est obtenu à partir de la spécialisation du concept *Destination* en *RuralArea* du motif S_1. Le même motif, S, s'obtient par la spécialisation du concept *Adventure* en *Safari* dans le motif S_2. Finalement, S_3 permet d'obtenir S en spécialisant la relation *hasActivity* en *hasAdventure*.

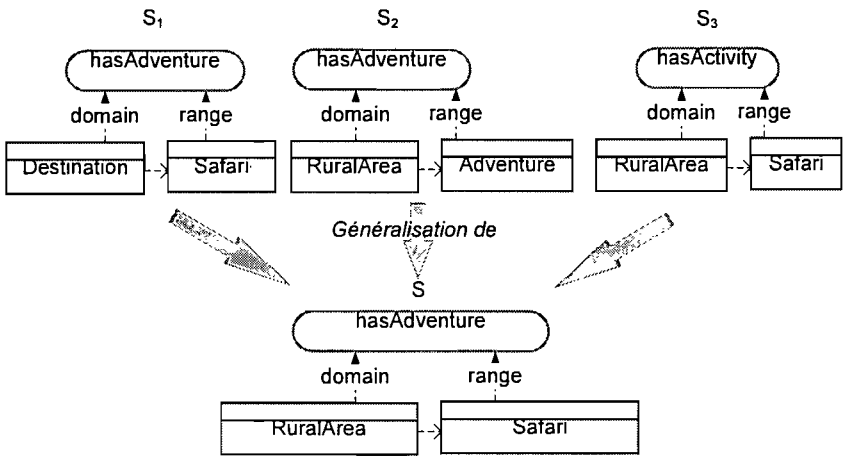


Figure 5.2 – Génération redondante par l'application d'opérations élémentaires.

Afin de réduire davantage la redondance lors de la génération des motifs, nous ajoutons de nouvelles contraintes aux opérations de spécialisation des motifs qui fixent l'endroit d'application des opérations élémentaires.

5.1.3 Opérations canoniques

Idéalement, il doit y avoir un seul générateur prédécesseur canonique. Pour le moment, notre objectif est plus modeste, et consiste à restreindre l'endroit de la séquence où une opération sera appliquée. Ainsi, cela nous permet de réduire le nombre de redondances dans la génération des motifs. En effet, nous avons décidé d'autoriser l'ap-

plication des opérations à un endroit unique qui est la fin de la séquence. Plus spécifiquement, seulement les classes finales seront éligibles pour la spécialisation, alors que les nouvelles classes peuvent être ajoutées seulement à la fin de la séquence. Concernant les relations, ceci signifie que la classe cible d'un triplé ajouté/spécialisé, doit être nécessairement la classe finale de la séquence.

Les opérations résultantes, appelées *canoniques*, sont le produit de la restriction des opérations élémentaires correspondantes. Elles sont nommées par le remplacement du suffixe *"E"* par *"C"* aux noms des opérations élémentaires : $addClsC(), splClsC()$, $addRelC()$ and $splRelC()$. De plus, à la suite de cette restriction, le dernier argument peut être systématiquement omis, puisque la position de l'application de l'opération est maintenant fixée à la fin de la séquence.
Ainsi, les nouvelles classes sont ajoutées à la fin de la séquence tel que donné par la définition 5.1.11 ci-dessous.

Définition 5.1.11. ***Ajout canonique de classe***

Soient $S \in \Gamma_\Omega$, $c \in \mathscr{C}_\Omega$. *L'opération* $addClsC(S,c)$ *est définie par :* $addClsC(S,c) = addClsE(S,c,|S.\varsigma|+1)$

Par exemple : $addClsC((\langle Beach, Sports\rangle, \{hasActivity(1,2)\}), Accommodation) = (\langle Beach, Sports, Accommodation\rangle, \{hasActivity(1,2)\})$.

Pour la spécialisation canonique d'un concept, seulement les remplacements à la position finale sont permis.

Définition 5.1.12. ***Spécialisation canonique de classe***

Soient $S \in \Gamma_\Omega$ *et* $c \in \mathscr{C}_\Omega$. *L'opération* $splClsC(S,c)$ *est définie par :* $splClsC(S,c) = splClsE(S,c,|S.\varsigma|)$

À titre d'exemple nous avons : $splClsC((\langle Beach, Sports\rangle, \{\}), BeachVolley) = (\langle Beach, BeachVolley\rangle, \{\})$.

De même, l'insertion d'une nouvelle relation dans une séquence sera autorisée seulement si la classe destination est la dernière classe.

Définition 5.1.13. ***Ajout canonique de relation***

Soient $S \in \Gamma_\Omega$ *et* $r \in \mathscr{R}_\Omega$. *L'opération* $addRelC(S, r, i)$ *est définie par :* $addRelC(S, r, i) = addRelE(S, r, i, |S.\varsigma|)$

Par exemple : $addRelC((\langle Beach, BeachVolley\rangle, \{\}), hasActivity, 1) = (\langle Beach, BeachVolley\rangle, \{hasActivity(1,2)\})$.

La spécialisation d'une relation concernera seulement les relations d'une séquence qui ont la dernière classe dans le co-domaine (voir la définition 5.1.14).

Définition 5.1.14. ***Spécialisation canonique de relation***

Soient $S \in \Gamma_\Omega$ *et* $r, r' \in (\mathscr{R}_\Omega)^2$. *L'opération* $splRelC(S, r, i, r')$ *est définie par :* $splRelC(S, r, i, r') = splRelE(S, r, i, |S.\varsigma|, r')$

Par exemple : $splRelE((\langle Beach, Golf\rangle, \{hasActivity(1,2)\}), hasActivity, 1, hasSports) = (\langle Beach, Golf\rangle, \{hasSports(1,2)\})$.

Le pseudo-code de ces quatre opérations canoniques est présenté dans les algorithmes 11, 12, 13, et 14. Ces algorithmes génèrent des motifs d'un niveau $k+1$ par l'application des opérations canoniques sur le motif de niveau k qui est transmis en tant que paramètre.

5.1.4 *xPMiner* : algorithme à la *Apriori* pour la recherche de motifs fréquent

Pour retrouver la famille de motifs fréquents $\mathscr{F}_\sigma$, nous proposons un algorithme qui combine l'exploitation du principe de l'algorithme *Apriori* et l'utilisation des opérations canoniques. L'algorithme, appelé *xPMiner* adopte ainsi une descente par niveau de l'espace $\langle \Gamma_\Omega, \sqsubseteq_{\Gamma_\Omega} \rangle$ similaire à celle d'*Apriori*. Plus précisément, au niveau $k+1$, *xPMiner* utilise les motifs fréquents générés au niveau k pour identifier les motifs candidats de rang $k+1$ (appelés aussi *(k+1)-candidats*) par l'application des opérations canoniques aux motifs de niveau k (appelés aussi *k-fréquents*). Par la suite, la fréquence de chaque candidat est calculée afin de déterminer ceux qui sont fréquents.

Algorithm 9 xPMiner

Entrée :
Ω; ▷ Ontologie du domaine
$\mathscr{D}$, σ ; ▷ Ensemble de séquences d'objets étendues et seuil minimal du support

Sortie :
$\mathscr{F}_\sigma$; ▷ Ensemble de motifs fréquents

Initialisation :
$\mathscr{D}_\Omega \leftarrow$ ISEQTRANS($\mathscr{D}$) ;
$\mathscr{F}_c^0 \leftarrow \{\emptyset_{\Gamma_\Omega}\}$;

Méthode :
for $(k = 1; \mathscr{F}_\sigma^{k-1} \neq \emptyset; k++)$ **do**
    $\mathscr{F}_c^k \leftarrow \emptyset$;
    for all $S \in \mathscr{F}_\sigma^{k-1}$ **do**
        $\mathscr{F}_c^k \leftarrow \mathscr{F}_c^k \cup$ ADDCLSCN(S) $\cup$ SPLCLSCN(S) $\cup$ ADDRELCN(S) $\cup$ SPLRELCN(S)
    end for
    $\mathscr{F}_\sigma^k \leftarrow$ CANDTEST($\mathscr{F}_c^k, \mathscr{D}_\Omega, \sigma$) ;
end for
$\mathscr{F}_\sigma \leftarrow \bigcup_{i=1,k-1} \mathscr{F}_\sigma^i$;
return $\mathscr{F}_\sigma$;

Durant la phase d'initialisation, *xPMiner* transforme les séquences d'objets de départ

Algorithm 10 Tester les candidats

```
1: procedure CANDTEST(P : ensemble de motifs, 𝒟_Ω : séquences étendues, σ : support minimal)
2:     ℱ_temp ← ∅ ;
3:     for all S ∈ P do
4:         count ← 0 ;
5:         for all s ∈ 𝒟_Ω do
6:             if INST(s,S) then
7:                 count + + ;
8:             end if
9:         end for
10:        if count ≥ σ then
11:            ℱ_temp ← ℱ_temp ∪ {S} ;
12:        end if
13:    end for
14:    return ℱ_temp ;
15: end procedure
```

Algorithm 11 Génération de candidats par l'ajout d'une classe

```
1: procedure ADDCLSCN(S : motif, taille_max)
2:     k ← |S.ς| ;
3:     ℱ_temp ← ∅ ;
4:     for all c ∈ 𝒞_Ω : P_c(c) = 1 do
5:         S.ς[k+1] ← c;
6:         ℱ_temp ← ℱ_temp ∪ {S} ;
7:     end for
8:     return ℱ_temp;
9: end procedure
```

Algorithm 12 Génération de candidats par l'ajout d'une relation

procedure ADDRELCN(S : motif)
  $k \leftarrow |S.\varsigma|$;
  $\mathscr{F}_{temp} \leftarrow \emptyset$;
  for all $r \in \mathscr{R}_\Omega : P_r(r) = 1$ tel que $S.\varsigma[k] \ll ran(r)$ **do**
    for all $0 < i \leq k$ tel que $S.\varsigma[k] \ll ran(r, S.\varsigma[i])$ **do**
      if $r(i,k) \notin S.\theta$ **then**
        $\mathscr{F}_{temp} \leftarrow \mathscr{F}_{temp} \cup \{(S.\varsigma, S.\theta \cup \{r(i,k)\})\}$;
      end if
    end for
  end for
  return $\mathscr{F}_{temp}$;
end procedure

Algorithm 13 Génération de candidats par la spécialisation d'une classe

procedure SPLCLSCN(S : motif)
  $k \leftarrow |S.\varsigma|$;
  $\mathscr{F}_{temp} \leftarrow \emptyset$;
  for all $c \in \mathscr{C}_\Omega : c \sqsubseteq^r_\Omega S.\varsigma[k]$ **do**
    if $\forall r(i,j) \in S.\theta : c \ll ran(r, S.\varsigma[i])$ **then**
      $S.\varsigma[k] \leftarrow c$;
      $\mathscr{F}_{temp} \leftarrow \mathscr{F}_{temp} \cup \{S\}$;
    end if
  end for
  return $\mathscr{F}_{temp}$;
end procedure

Algorithm 14 Génération de candidats par la spécialisation d'une relation

procedure SPLRELCN(S : motif)
  $k \leftarrow |S.\varsigma|$;
  $\mathscr{F}_{temp} \leftarrow \emptyset$;
  for all $r(i,k) \in S.\theta$ **do**
    for all $r' \in \mathscr{R}_\Omega : r' \sqsubseteq^r_\Omega r$ **do**
      if $\nexists r'(i,k) \in S.\theta$ **then**
        $\mathscr{F}_{temp} \leftarrow \mathscr{F}_{temp} \cup \{(S.\varsigma, (S.\theta - \{r(i,k)\}) \cup \{r'(i,k)\})\}$;
      end if
    end for
  end for
  return $\mathscr{F}_{temp}$;
end procedure

de $\mathscr{D}$ en séquences d'objets étendues moyennant une procédure décrite par l'algorithme 2. Celle-ci ajoute les liens de l'ontologie qui connectent deux objets de la séquence. Par la suite, le motif vide est systématiquement ajouté à l'ensemble des motifs fréquents de niveau 0. Pour chaque niveau $k+1$ ($k \geq 0$), les candidats sont générés à partir des motifs fréquents du niveau k par l'application des opérations canoniques. Ceci tranche avec la pratique de combiner les motifs de rang k qui est utilisée par *Apriori* (lignes 10 à 16). Dans une deuxième étape, les candidats sont testés avec la base de séquences d'objets étendues $\mathscr{D}_\Omega$ pour ne retenir que les motifs se manifestant un nombre de fois supérieur ou égal au seuil fixé (ligne 15). L'algorithme s'arrête lorsque l'ensemble des motifs candidats d'un niveau est vide.

5.1.5 Complétude de l'approche de fouille

Comme nous venons de le voir, l'algorithme *xPMiner* recherche les motifs fréquents en parcourant l'espace des motifs par niveau, où chaque niveau est composé de motifs ayant le même niveau d'abstraction. De plus, les opérations canoniques assurent une descente progressive dans l'espace des motifs de sorte à ce que l'application d'une opération canonique à un motif de niveau k engendre un motif de niveau $k+1$. Il reste maintenant à savoir si *xPMiner* génère tous les motifs fréquents d'un espace donné.

Pour démontrer la complétude de l'algorithme *xPMiner*, nous démontrons ce qui suit.

- Tous les motifs fréquents d'un niveau peuvent être générés à partir des motifs du niveau qui le précède en appliquant les opérations canoniques ;
- Suivant le principe d'élagage de *xPMiner* aucun motif fréquent n'est écarté et tous les motifs découverts sont fréquents.

Alors que la première partie de la démonstration est donnée par la proposition 5.1.2, la deuxième partie suit la structure de l'algorithme *xPMiner*. Il est clair qu'aucun motif non fréquent n'est généré par l'algorithme. En effet, chaque motif candidat est soumis au test de fréquence (voir l'algorithme 10), et un motif est automatiquement éliminé s'il s'avère non fréquent.

Il nous reste à démontrer qu'aucun motif fréquent n'est écarté par le principe d'élagage de l'algorithme *xPMiner*.
Pour arriver à cette fin, nous procédons par l'absurde. Notre hypothèse de départ est qu'il existe un motif S_1, fréquent de niveau $k+1$ qui peut être généré à partir du motif non fréquent S_2 de rang k par l'application d'une opération canonique. Comme $S_1 \sqsubseteq_{\Gamma_\Omega} S_2$, alors $[S_1]_{\Gamma_\Omega} \subseteq [S_2]_{\Gamma_\Omega}$. Ce qui signifie que le motif S_2 couvre au moins les séquences d'objets que S_1 couvre. D'où la fréquence de S_2 est au moins égale à la fréquence de S_1. Étant donné que S_1 est fréquent alors S_2 l'est aussi. Ce qui contredit l'hypothèse de départ.
Ainsi, nous concluons qu'avec les opérations canoniques, un motif fréquent ne peut être généré à partir d'un motif non fréquent du niveau qui le précède.

La proposition 5.1.2 ci-dessous stipule que si un motif d'un rang donné existe, alors il est possible de le générer à partir d'un motif du niveau qui le précède par l'application d'une opération canonique.

Proposition 5.1.2. ***Complétude de l'ensemble des opérations canoniques***

Pour un nombre arbitraire k, si les motifs de rang $k+1$ existent, alors chacun de ces motifs peut être généré par l'application des opérations canoniques à des motifs de rang k.

La démonstration de la proposition sur la complétude de l'ensemble des opérations canoniques est présentée dans l'annexe I.8.

5.2 Optimisation du processus de fouille

Dans la sous-section 5.1.3, nous avons présenté les opérations canoniques pouvant être utilisées pour calculer la subsomption entre un motif de rang k et un motif de rang $k+1$. Ces opérations sont le résultat de l'ajout de contraintes sur les opérations générales de spécialisation de motif (voir la sous-section 4.2.7.1). Elles nous permettent d'explorer l'espace $\langle \Gamma_\Omega, \sqsubseteq_{\Gamma_\Omega} \rangle$ de manière homogène et de réduire la redondance lors de

la génération des motifs d'un niveau donné. Cependant, la redondance des motifs n'est pas complètement éliminée. Par exemple, la figure 5.3 montre un motif de rang égal à 6 qui est obtenu par l'application de deux opérations canoniques sur deux motifs distincts de rang 5. La première opération, qui consiste en la spécialisation canonique de concept, est appliquée au motif du côté haut gauche de l'image, et consiste à remplacer le concept *Adventure* par le concept *Safari*. La deuxième opération, qui est la spécialisation canonique de relation, remplace dans le motif du côté haut droit la relation *hasActivity* par la relation *hasAdventure*.

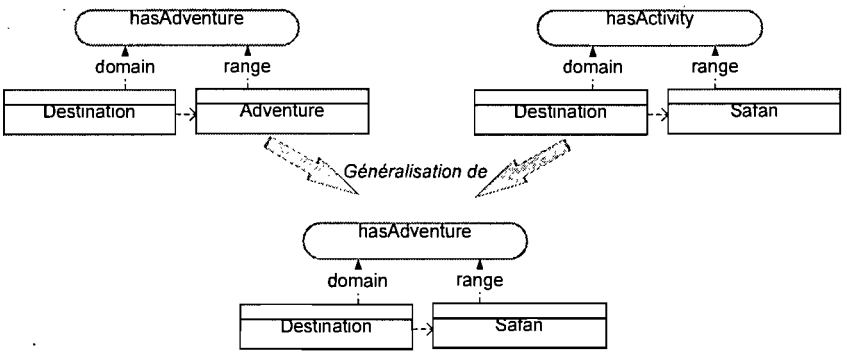


Figure 5.3 – Génération redondante par l'application d'opérations canoniques.

Comme on vient de le voir, à un niveau k, plus d'un motif peut conduire à la génération d'un même motif de niveau $k+1$ par l'application de différentes opérations canoniques. Notre objectif est d'éliminer toute génération redondante de motifs dans le processus de fouille. Ainsi, pour chaque motif, il y n'aura qu'un seul chemin de génération. Afin d'arriver à cet objectif, nous considérons un chemin inverse de celui qui nous permet d'avoir une décomposition canonique et unique d'un motif, c'est-à-dire, le réduire vers le motif vide.

La figure 5.4 est une illustration de l'intuition qui vient d'être présentée. Le motif $(\langle Destination, Safari\rangle, \{hasAdventure(1,2)\})$ est défait de manière canonique jusqu'à l'obtention du motif $\emptyset_{\Gamma_\Omega}$. Par la suite, le motif de départ est reconstitué en suivant le chemin inverse, et en ayant comme point de départ $\emptyset_{\Gamma_\Omega}$. Ce chemin correspond à l'application des opérations canoniques duales aux opérations appliquées lors de la décom-

position.

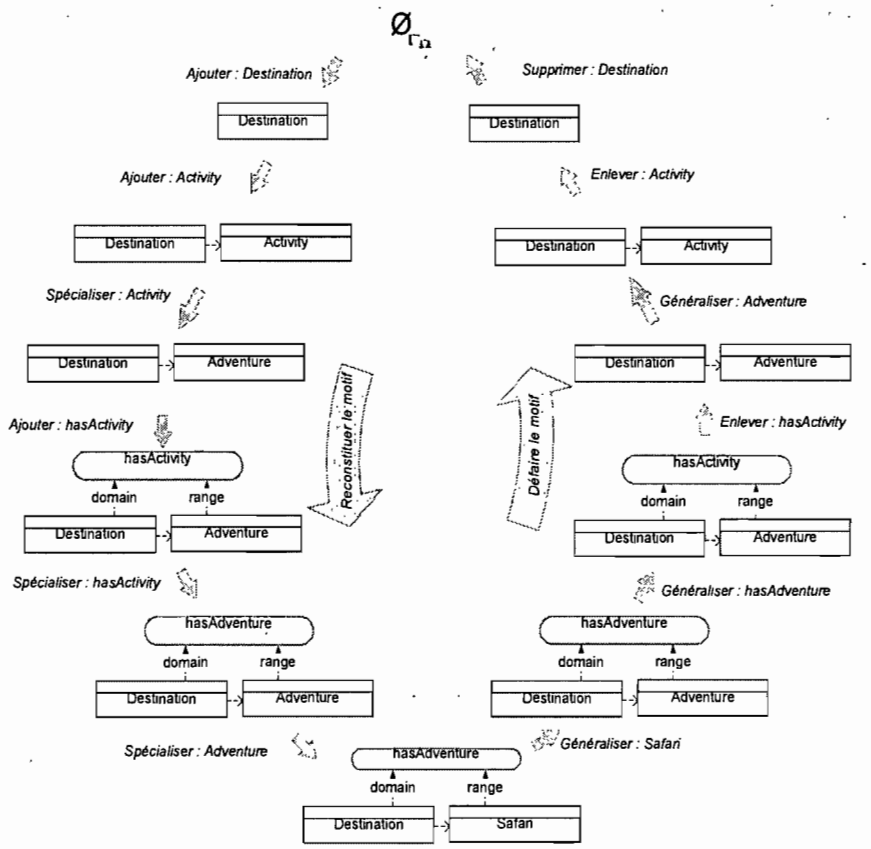


Figure 5.4 – Exemple d'un motif qui est défait et reconstitué en suivant le chemin inverse.

Cependant, à un moment donné, on ne sait pas encore si l'opération courante est impliquée dans le chemin de génération unique. Admettons que nous ayons un motif S à spécialiser en S' par une opération canonique qui fait introduire l'élément A. Ce dernier peut être soit un concept ou un triplet de la forme $(S.\varsigma[l], r, S.\varsigma[m])$ qui est simplifié à $r(l,m)$ où l et m sont des nombres naturels qui dénotent, respectivement, les positions de deux concepts source et destination de r dans $S.\varsigma$.

Pour garantir une génération unique, il faudrait s'assurer que l'opération canonique qui fait introduire A est la plus prioritaire parmi les autres opérations qui peuvent conduire

à la production de S' et qui font intervenir des éléments autres que A. Pour cela, nous calculons les autres éléments qui seront introduits par ces opérations, et on s'assure qu'il n'y a aucun élément dans cet ensemble, appelé *ensemble critique* (voir les définitions 5.2.3, 5.2.4 et 5.2.5), qui *est plus grand* que A.

Les grandes lignes de l'ordre total sur les concepts et les triplets, que nous proposons dans la définition 5.2.1, sont comme suit. Les concepts sont toujours considérés comme étant plus petits que les triplets. Lorsqu'on compare deux concepts, on se base sur l'ordre lexicographique des URIs (libellés dans l'ontologie) correspondantes. La comparaison des triplets, quant à elle, se base non seulement sur l'ordre lexicographique des URIs, mais aussi sur les bornes du triplet (positions des concepts auxquels la relation du triplet est attachée). Ainsi, si deux triplets font intervenir des relations ayant des URIs identiques, on compare les bornes inférieures qui correspondent aux positions des concepts sources dans le motif en question, et si les bornes inférieures sont égales, une comparaison des bornes supérieures (positions des concepts destination des relations dans le motif) est effectuée. De manière plus rigoureuse, l'ordre total sur l'ensemble des éléments d'un motif est donné par la définition ci-dessous 5.2.1.

Définition 5.2.1. ***Ordre total sur l'ensemble des concepts et triplets***

Soient l'ontologie Ω, $\mathscr{C}_\Omega$ *(resp.* $\mathscr{R}_\Omega$*) l'ensemble des concepts (resp. des relations) de* Ω, ρ *l'ensemble des triplets de* $\mathscr{R}_\Omega \times \mathbb{N} \times \mathbb{N}$, *et* $\mathscr{L}$ *une fonction qui retourne l'URI d'un concept ou d'une relation. L'ordre lexicographique de deux chaînes de caractères est désigné par les relations* $\leq_{lex}$, $<_{lex}$ *et* $=_{lex}$.

La relation d'ordre total sur les éléments de $\mathscr{C}_\Omega \cup \rho$ *est définie comme suit. Soient A et B deux éléments de* $\mathscr{C}_\Omega \cup \rho$. *On dit que* $A \leq B$ *si et seulement si l'une des conditions suivantes se présente :*

1. $A \in \mathscr{C}_\Omega \wedge B \in \rho$,
2. $(A,B) \in (\mathscr{C}_\Omega)^2 : \mathscr{L}(A) \leq_{lex} \mathscr{L}(B)$,
3. $(A,B) \in (\mathscr{R}_\Omega)^2$ *avec* $A = r_1(l_1,m_1), B = r_2(l_2,m_2)$ *tel que :* $(\mathscr{L}(r_1) <_{lex} \mathscr{L}(r_2))$ *ou* $((\mathscr{L}(r_1) =_{lex} \mathscr{L}(r_2)) \wedge ((l_1 = l_2 \wedge m_1 \leq m_2) \vee (l_1 < l_2)))$.

Par exemple, nous avons $hasActivity(3,5) \leq hasAccommodation(1,2)$ et $Activity \leq Accommodation$ puisque $\mathscr{L}(hasActivity) <_{lex} \mathscr{L}(hasAccommodation)$ et $\mathscr{L}(Activity) <_{lex} \mathscr{L}(Accommodation)$.
Nous avons aussi $hasAccommodation(2,7) \leq hasAccommodation(3,4)$ puisque $2 < 3$.

Remarque 5.2.1. *Soient A, B deux éléments de $\mathscr{C}_\Omega \cup \rho$. On dit que $A = B$ si et seulement si l'une des conditions suivantes est satisfaite :*

- $(A,B) \in (\mathscr{C}_\Omega)^2 : \mathscr{L}(A) =_{lex} \mathscr{L}(B)$,
- $(A,B) \in (\mathscr{R}_\Omega)^2$ *avec* $A = r_1(l_1,m_1), B = r_2(l_2,m_2)$ *tel que :*
  - $\mathscr{L}(r_1) =_{lex} \mathscr{L}(r_2)$,
  - $l_1 = l_2$,
  - $m_1 = m_2$.

Suivant la proposition 5.2.1, la relation d'ordre $\leq$, définie sur l'ensemble des concepts et des triples liés à une ontologie, est totale. En d'autres mots, tout concept/triplet peut être comparé avec n'importe quel autre concept/triplet en utilisant la relation $\leq$ présentée dans la définition 5.2.1.

Proposition 5.2.1. ***Relation d'ordre total***

La relation $\leq$ est un ordre total sur l'ensemble $\mathscr{C}_\Omega \cup \rho$. Autrement dit, $\forall A, B \in (\mathscr{C}_\Omega \cup \rho)^2 : A \leq B \vee B \leq A$.

La démonstration de la proposition ci-dessus est présentée dans l'annexe I.9.

À partir de la relation d'ordre total $\leq$, définie sur l'ensemble $\mathscr{C}_\Omega \cup \rho$, nous définissons une relation d'ordre strict, dénotée par $<$, comme suit.

Définition 5.2.2. *Soient A et B deux éléments de* $\mathscr{C}_\Omega \cup \rho$. *On dit que A est strictement inférieur à B, noté* $A < B$, *si et seulement si :*

$$(A \leq B) \wedge (B \neq A).$$

Dans la suite, nous allons principalement exploiter la relation d'ordre total sur les concepts et triplets pour la comparaison d'un concept/triplet avec les éléments d'un ensemble critique donné. Les ensembles critiques sont déterminés par le motif impliqué et l'élément à introduire dans ce motif.

5.2.0.1 Ensembles critiques

Soient S un motif et A un élément (concept/triplet) qui est candidat potentiel à être introduit de manière canonique dans S pour générer S'. Suivant la nature et la profondeur de A, les cas à distinguer sont comme suit.

- A est un concept avec une profondeur égale à 1,
- A est un concept avec une profondeur supérieure à 1,
- A est un triplet dont la relation est d'une profondeur égale à 1,
- A est un triplet dont la relation est d'une profondeur supérieure à 1.

Pour chacun de ces cas, nous définissons les éléments de l'ensemble critique associé. Dans le premier cas, lorsque l'élément A est un concept de profondeur 1, l'opération associée est l'ajout de concept. Comme le motif qui a le concept A à la dernière position ne peut être obtenu que d'une seule façon (application de l'opération canonique d'ajout du concept A à S), alors l'ensemble critique désigné par $\Psi_c^1(A,S)$ est vide ($\Psi_c^1(A,S) = \emptyset$). En effet, aucune autre opération ne peut conduire à la génération du même motif S'.

Dans le deuxième cas, l'élément A est un concept de profondeur supérieure à 1. L'opération associée à l'introduction de cet élément dans S est la spécialisation canonique du dernier concept. Cependant, plusieurs opérations canoniques de spécialisation de concept ou d'ajout/spécialisation de relation peuvent amener à la génération du même

motif.

En effet, si A possède plusieurs prédécesseurs réguliers (*héritage multiple*), la spécialisation de chacun de ces concepts en A dans des motifs différents de S donnera lieu au motif S'. Ainsi, nous incluons dans l'ensemble critique de A tous ses prédécesseurs réguliers qui sont différents du dernier concept de S.

De plus, si des triplets de S ont comme destination le dernier concept de ce motif, d'autres opérations qui impliquent ces triplets peuvent conduire à la génération du même motif S'. C'est pour cette raison que les triplets qui font intervenir la dernière position sont ajoutés à l'ensemble critique.

L'opération canonique qui permet de générer S' par l'introduction de A ne sera autorisée que si A est plus grand que tous les éléments de l'ensemble critique sous-jacent.

Formellement, cet ensemble se définit comme suit.

Définition 5.2.3. ***Ensemble critique d'un concept non racine***

Soient Ω une ontologie, S une séquence de classes étendue de Γ_Ω, et c un concept de $\mathscr{C}_\Omega$, tel que $P_c(c) > 1$. L'ensemble critique de c par rapport à S est donné par la fonction $\Psi_c : \mathscr{C}_\Omega \times \Gamma_\Omega \rightarrow E(\mathscr{C}_\Omega \cup \rho)$, où $E(\mathscr{C}_\Omega \cup \rho)$ représente l'ensemble des parties de $\mathscr{C}_\Omega \cup \rho$, tel que :

$$\Psi_c(c,S) = \begin{cases} \{c' \in (\mathscr{C}_\Omega - \{S.\varsigma[|S.\varsigma|]\}) | c \sqsubseteq^r_\Omega c' \wedge \forall r(i,|S.\varsigma|) \in S.\theta : c' \ll ran(r,S.\varsigma[i])\} \\ \cup \\ \{r'(i,|S.\varsigma|) \in S.\theta|\} \end{cases}$$

Par exemple, l'ensemble critique du concept *Safari* de l'ontologie *Travel* par rapport au motif $S = (\langle Destination, Adventure\rangle, \{hasActivity(1,2)\})$, comporte *Sightseeing* qui est le deuxième parent régulier de *Safari*, et le triplet $hasActivity(1,2)$ qui et le seul triplet de S où la dernière position (2) est impliquée. Autrement dit, nous avons $\Psi_c(Safari,S) = \{Sightseeing, hasActivity(1,2)\}$.

Nous constatons que le concept *Sightseeing* pourrait être impliqué dans une opération de spécialisation canonique de concept pour générer le même motif que celui généré par

la spécialisation de *Adventure* en *Safari* dans S.
En effet, nous avons $splClsC(S, Safari) = splClsC((\langle Destination, Sighseeing\rangle, \{hasActivity(1,2)\}), Safari)$. D'autre part, le même motif $((\langle Destination, Safari\rangle, \{hasActivity(1,2)\})$ peut être obtenu du motif $(\langle Destination, Safari\rangle, \{\})$ par l'ajout canonique de la relation *hasActivity* aux positions 1 et 2. C'est pour cette raison que le triplet $hasActivity(1,2)$ figure dans l'ensemble critique $\Psi_c(Safari, S)$.

Dans le troisième cas, l'élément A est un triplet, noté $r(i, |S.\varsigma|)$ où $P_r(r) = 1$. L'opération associée à l'introduction de cet élément dans S est l'opération canonique d'ajout de relation.
Le calcul de l'ensemble critique sous-jacent passe par la réponse à la question suivante : Quels sont les autres éléments qui, impliqués dans d'autres opérations canoniques, donneront lieu au même motif S' obtenu par l'ajout de A à S ?
La réponse à cette question est comme suit. Si d'autres triplets de S ont comme destination le dernier concept de ce motif, et que les relations associées sont de profondeur 1, d'autres opérations d'ajout canonique de relation qui impliquent ces triplets peuvent conduire à la génération de S'. Ces triplets sont donc considérés comme étant des éléments de l'ensemble critique.
Étant donné que pour la génération d'un même motif, nous considérons les opérations qui font intervenir des relations racines plus prioritaires que les opérations qui font intervenir des relations non racines, les triplets avec relations non racines ne sont pas inclus dans l'ensemble critique.
Pour la même raison, les concepts ne font pas partie de l'ensemble critique. La définition formelle de cet ensemble est donnée ci-dessous.

Définition 5.2.4. ***Ensemble critique d'un triplet composé d'une relation racine***

Soient Ω une ontologie, S une séquence de classes étendue de Γ_Ω, et $r(i, |S.\varsigma|)$ un triplet de ρ. L'ensemble critique de $r(i, |S.\varsigma|)$ par rapport à S, où $P_r(r) = 1$, est donné par la fonction $\Psi_r^1 : \Gamma_\Omega \times \rho \rightarrow E(\rho)$, où $E(\rho)$ représente l'ensemble des parties de ρ. Ψ_r se définit comme suit :

$$\Psi_r(r(i,|S.\varsigma|),S) = \{r'(i',|S.\varsigma|) \in S.\theta - r(i,|S.\varsigma|) | P_r(r') = 1\}$$

À titre d'exemple, l'ensemble critique du triplet $hasActivity(1,2)$ par rapport au motif $S = (\langle Destination, Safari\rangle, \{hasAdventure(1,2)\})$, est $\Psi_r^1(hasActivity, S) = \emptyset$. En effet, le motif S ne contient aucune relation racine qui aboutit à la dernière position.

Dans le dernier cas, l'élément A est un triplet, noté $r(i,|S.\varsigma|)$, et r n'est pas une relation racine. L'opération associée à l'introduction de cet élément dans S est la spécialisation canonique d'une relation. Étant donné que les opérations de spécialisation de relation sont considérées plus prioritaires que les opérations de spécialisation de concepts pouvant mener à la génération d'un même motif, seulement les triplets dont les relations ne sont pas des racines sont considérées. Il est à noter que les relations racine, qui ne font pas partie de l'ensemble critique, sont considérées au niveau de l'opération de suffixe de spécialisation de relation (voir la définition 5.2.9). C'est ainsi que seulement les triplets de $S.\theta$ qui peuvent être le résultat d'une spécialisation de triplets d'autres motifs pouvant conduire à la génération de S' sont considérés. Les deux cas suivant sont à distinguer.

1. S possède d'autres triplets avec les positions i et $|S.\varsigma|$, et dont les relations sont des successeurs réguliers de la relation de S qui se spécialise en r. Ainsi, S' pourrait être obtenu par la spécialisation d'une de ces relations en r.
2. S possède d'autres relations non racines faisant intervenir la dernière position dont au moins un parent régulier n'est pas dans S et qui est différent de r. Dans ce cas, S' pourrait être obtenu à partir d'un autre motif par la spécialisation d'une relation en une relation non racine de S ;

Les triplets couverts par ces cas constituent l'ensemble critique. Il est à noter que lors du calcul de l'ensemble critique, on veille à ne pas inclure les deux triplets du cas de la spécialisation courante. Formellement cet ensemble se définit comme suit.

Définition 5.2.5. ***Ensemble critique d'un triplet ayant une relation non racine***

Soient Ω une ontologie, S une séquence de classes étendue de Γ_Ω, et $r(i,|S.\varsigma|)$ un triplet de ρ tel que $P_r(r) > 1$. L'ensemble critique de $r(i,|S.\varsigma|)$ par rapport au triplet $r'(i,|S.\varsigma|)$ de S est donné par la fonction $\Psi_r^n : \rho \times \rho \times \Gamma_\Omega \rightarrow E(\rho)$ [2] tel que $r''(i',|S.\varsigma|) \in \Psi_r^n(r(i,|S.\varsigma|), r'(i,|S.\varsigma|), S)$, si et seulement si :

- $r''(i',|S.\varsigma|) \in S.\theta - \{r(i,|S.\varsigma|), r'(i,|S.\varsigma|)\}$;
- $(r'' \sqsubseteq_\Omega^r r'$ *et* $i' = i)$ *ou* $(\exists r'''(i',|S.\varsigma|) \in \rho - S.\theta - \{r(i,|S.\varsigma|)\}, r'' \sqsubseteq_\Omega^r r''')$.

Par exemple, l'ensemble critique du triplet $hasSafari(1,2)$ par rapport au triplet $hasAdventure(1,2)$ du motif $S = (\langle Destination, Safari\rangle, \{hasAdventure(1,2)\})$, est composé de $hasSightseeing(1,2)\}$ qui est le seul triplet dont la relation est parent régulier de $hasSafari$ et qui ne se trouve pas dans $S.\theta$. Autrement dit, $\Psi_r^n(hasSafari(1,2), hasAdventure(1,2), S) = \{hasSightseeing\ (1,2)\}$. Ainsi, le triplet $hasSightseeing(1,2)\}$ pourrait être impliqué dans une opération de spécialisation canonique de relation pour générer le même motif que celui généré par la spécialisation de $hasAdventure(1,2)$ en $hasSafari(1,2)$ dans S. En effet, nous avons $splRelC(S, hasAdventure, 1, hasSafari) = splRelC((\langle Destination, Safari\rangle, \{hasSightseeing(1,2)\}), hasSightseeing, 1, hasSafari)$.

5.2.0.2 Opérations de suffixe

En se basant sur les relations d'ordre (voir les définitions 5.2.1 et 5.2.2), sur les ensembles critiques, et sur les opérations canoniques définies dans la section 5.1.3, nous définissons un ensemble de quatre opérations, appelées opérations de suffixe. Ces opérations permettent de générer des motifs sans redondance.

L'opération de suffixe pour l'ajout de concept est équivalente à l'opération canonique d'ajout de concept. Il s'agit d'ajouter un concept de haut niveau d'abstraction à la fin d'un motif (voir la définition 5.2.6).

Définition 5.2.6. ***Opération de suffixe pour l'ajout de concept***

2. $E(\rho)$ représente l'ensemble des parties de ρ

Soient $S \in \Gamma_\Omega$ et $c \in \mathscr{C}_\Omega$ tel que $P_c(c) = 1$. L'opération de suffixe pour l'ajout du concept c à la dernière position de S, dénotée $addClsA(S,c)$, est possible si et seulement si les conditions de l'application de l'opération d'ajout canonique de c est possible. Autrement dit, nous avons $addClsA(S,c) = addClsC(S,c)$.

Pour l'opération de suffixe de spécialisation de concept, les remplacements dans la position finale sont permis seulement si les conditions de la spécialisation canonique d'un concept sont vérifiées, et que le dernier concept est plus grand que tous les éléments de l'ensemble critique.
Cette dernière condition garantie que l'opération en question est plus prioritaire que toutes les autres opérations qui permettent de générer le même motif et où sont impliqués les éléments de l'ensemble critique.

Définition 5.2.7. ***Opération de suffixe pour la spécialisation de concept***

Soient $S \in \Gamma_\Omega$ et $c \in \mathscr{C}_\Omega$. L'opération de suffixe pour la spécialisation du dernier concept de S par c, dénotée par $splClsA(S,c)$, est possible si et seulement si les conditions de l'application de l'opération de spécialisation canonique de c est possible et que :

$$\forall A \in \Psi_c(c,S) : A < S.\varsigma[|S.\varsigma|]$$

Si ces conditions sont satisfaites, alors $splClsA(S,c) = splClsC(S,c)$.

L'opération de suffixe pour l'ajout d'une relation à un motif n'est possible que si les conditions de l'application de l'opération canonique d'ajout de relation sont satisfaites et que l'opération en question est la plus prioritaire de toutes les opérations qui peuvent mener à la génération du même motif par l'ajout canonique d'autres relations. Cette dernière condition permet de s'assurer que le triplet qui va résulter de l'ajout de la relation est le plus grand de tous les éléments de l'ensemble critique sous-jacent. Ainsi, on a la garantie que le motif résultant est généré en effectuant l'opération la plus prioritaire. En d'autres mots, nous avons ce qui suit.

Définition 5.2.8. ***Opération de suffixe pour l'ajout de relation***

Soient $S \in \Gamma_\Omega$ et $r \in \mathscr{R}_\Omega$. L'opération de suffixe pour l'ajout de la relation $r(i, |S.\varsigma|)$ à $S.\theta$, dénotée par $addRelA(S, r, i)$, est possible si et seulement si les conditions de l'application de l'opération de l'ajout canonique de relation sont satisfaites et que :

$$\forall A \in \Psi_r^1(r(i, |S.\varsigma|), S) : A < r(i, |S.\varsigma|)$$

Si ces conditions sont satisfaites, alors $addRelA(S, r, i) = addRelC(S, r, i)$

L'opération de suffixe pour la spécialisation de relation dans un motif S pour générer un autre motif S' est possible si toutes les conditions de l'application de l'opération de spécialisation canonique de relation sont satisfaites, et que la relation en question est la plus prioritaire de toutes les autres opérations qui mènent à la génération du même motif.

Une opération de spécialisation de relation est considérée prioritaire si les conditions ci-dessous sont satisfaites.

1. **Il n'existe aucune relation racine qui fait intervenir la dernière position de S :** si une telle situation se présente, l'opération canonique de l'ajout d'une telle relation à un motif qui permet de générer S' sera considérée plus prioritaire, et donc l'opération courante de spécialisation de S en S' ne sera pas autorisée.
2. **Le triplet associé à la relation remplaçante est le plus grand de tous les éléments de l'ensemble critique sous-jacent :** ceci nous garantie que les opérations de spécialisation où est impliquée la relation remplaçante sont plus prioritaires que toutes les autres opérations de spécialisation canonique où sont impliquées d'autres relations de S.
3. **La relation à remplacer est la plus grande de tous les parents réguliers de la relation remplaçante :** cette condition permet de garantir que parmi toutes

les opérations de spécialisation où est impliquée la relation remplaçante celle qui porte sur la relation courante à remplacer est la plus prioritaire.

De manière plus rigoureuse, nous avons ce qui suit.

Définition 5.2.9. ***Opération de suffixe pour la spécialisation de relation***

Soient $S \in \Gamma_\Omega$ et $r \in \mathscr{R}_\Omega$. L'opération de suffixe pour la spécialisation de la relation $r'(i,|S.\varsigma|)$ par $r(i,|S.\varsigma|)$ dans le motif S, dénotée par $splRelA(S,r',r,i)$ est possible si et seulement si les conditions de l'application de l'opération de spécialisation canonique de relation sont satisfaites et que :

1. $\forall r''(i',|S.\varsigma|) \in S.\theta - \{r'(i,|S.\varsigma|)\}, P_c(r'') > 1$;
2. $\forall A \in \Psi_r^n(r(i,|S.\varsigma|),r'(i,|S.\varsigma|),S) : A < r(i,|S.\varsigma|)$;
3. $\forall r''(i,|S.\varsigma|) \in \rho - S.\theta$, *si* $r \sqsubseteq_\Omega^r r''$ *alors* $r''(i,|S.\varsigma|) < r'(i,|S.\varsigma|)$.

Si ces conditions sont satisfaites, alors $splRelA(S,r',r,i) = splRelC(S,r',r,i)$

La figure 5.5 représente un exemple de construction de motifs par l'application des opérations de suffixe. En démarrant du motif vide $\emptyset_{\Gamma_\Omega}$, on applique des opérations de suffixe pour obtenir le motif $(\langle Destination, Safari\rangle, \{hasAdventure(1,2)\})$. Comme on peut le constater, deux branches se distinguent. L'exploration de la branche gauche de l'image permet l'obtention du motif $(\langle Destination, Safari\rangle, \{hasAdventure(1,2)\})$. Par contre, le développement de la deuxième sous-branche (du côté droit de l'image), ne permet maintenant plus d'obtenir ce motif. En effet, lorsqu'on compare le concept *Adventure* avec les éléments de l'ensemble critique $\Psi_c^n(Adventure, (\langle Destination, Activity\rangle, \{hasAdventure(1,2)\}))$, on trouve qu'il n'est pas le maxima, car la relation $hasActivity(1,2)$ qui fait partie de cet ensemble est plus grande que le concept *Activity*. D'où le rejet de la spécialisation du concept *Activity* en *Adventure*.

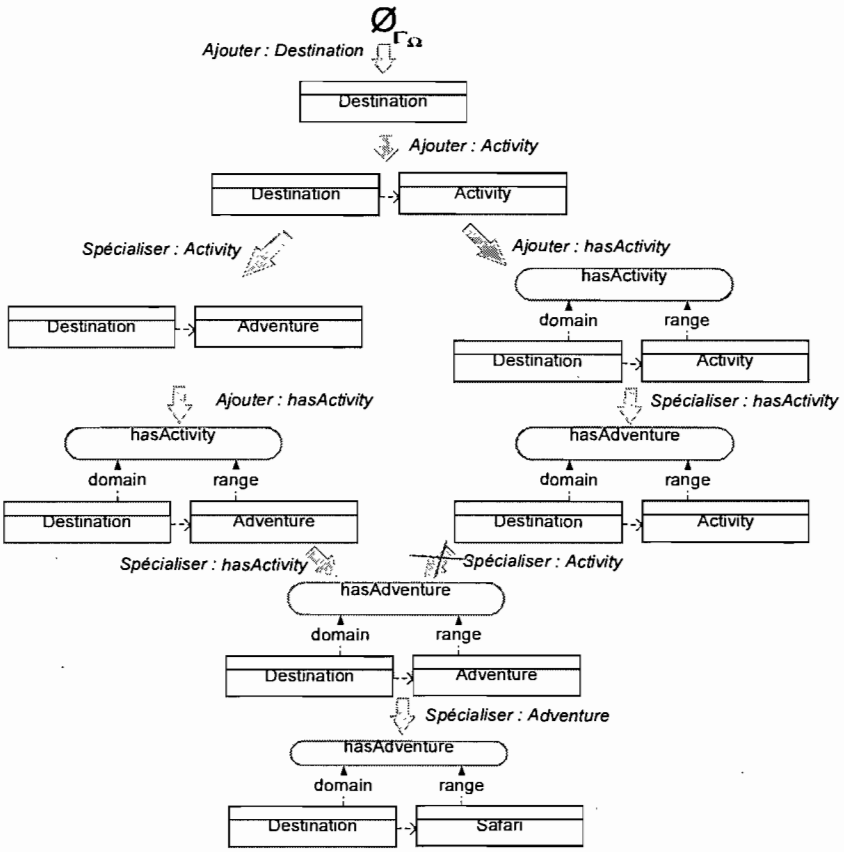


Figure 5.5 – Exemple de l'application des opérations de suffixe pour la génération de motifs.

5.2.0.3 Fouille de motifs fréquents basée sur les opérations de suffixe

Les opérations de suffixe présentées ci-dessus nous permettent de générer des motifs d'un niveau $k+1$ à partir de motifs du niveau k. L'exploitation de ces opérations dans un processus de fouille requiert que cet ensemble soit complet. Ceci veut dire qu'elles doivent garantir la génération de l'ensemble de tous les motifs d'un espace donné. En d'autres mots, chaque motif d'un niveau donné doit pouvoir être généré à partir d'un motif du niveau qui le précède. Cette complétude est justement le sujet de la proposition 5.2.2.

Proposition 5.2.2. ***Complétude de l'ensemble des opérations de suffixe***

Pour un nombre arbitraire k, tous les motifs de rang $k+1$ peuvent être générés par l'application des opérations de suffixe aux motifs de rang k.

La preuve de la proposition ci-dessus est présentée dans l'annexe I.10.

La proposition 5.2.2 garantit la complétude de l'ensemble des quatre opérations de suffixe. Cependant, notre premier objectif derrière leur élaboration est d'éliminer la redondance lors de la génération des motifs. C'est dans cette optique que la proposition 5.2.3 a été formulée. Cette proposition stipule que l'application des opérations de suffixe garantit qu'un motif donné ne sera généré qu'une seule fois.

Proposition 5.2.3. ***Génération non redondante d'un motif***

Pour un nombre arbitraire k, tout motif de rang $k+1$, s'il existe, peut être généré de manière unique par l'application d'une opération de suffixe à un motif de rang k.

La preuve de la proposition ci-dessus est présentée dans l'annexe I.11.

5.2.1 Algorithme de fouille basé sur les opérations de suffixe

De l'extension de l'algorithme *xPMiner* pour utiliser les opérations de suffixe résulte un algorithme de fouille des motifs où chaque motif n'est généré qu'une seule fois.

Cet algorithme, appelé *xPMiner2* (voir l'algorithme 15), remplace les procédures des opérations canoniques par l'ensemble des quatre procédures des opérations de suffixe pour la génération de candidats. Le pseudo-code de ces opérations est représenté par les algorithmes 16, 18 19 et 17.

Algorithm 15 xPMiner2

Entrée :
Ω ; ▷ Ontologie du domaine
$\mathscr{D}$, σ ; ▷ Ensemble de séquences d'objets étendues, seuil du support

Sortie :
$\mathscr{F}_\sigma$; ▷ Ensemble de motifs fréquents

Initialisation :
$\mathscr{D}_\Omega \leftarrow$ ISEQTRANS($\mathscr{D}$) ;
$\mathscr{F}_c^0 \leftarrow \{\emptyset_{\Gamma_\Omega}\}$;

Méthode :
for $(k = 1; \mathscr{F}_\sigma^{k-1} \neq \emptyset; k++)$ **do**
    $\mathscr{F}_c^k \leftarrow \emptyset$;
    for all $S \in \mathscr{F}_\sigma^{k-1}$ **do**
        $\mathscr{F}_c^k \leftarrow \mathscr{F}_c^k \cup$ ADDCLSAN(S) $\cup$ SPLCLSAN(S) $\cup$ ADDRELAN(S) $\cup$ SPLRELAN(S)
    end for
    $\mathscr{F}_\sigma^k \leftarrow$ CANDTEST($\mathscr{F}_c^k, \mathscr{D}_\Omega, \sigma$) ;
end for
$\mathscr{F}_\sigma \leftarrow \bigcup_{i=1,k-1} \mathscr{F}_\sigma^i$;
return $\mathscr{F}_\sigma$;

Algorithm 16 Génération de candidats par les opérations de suffixe d'ajout de classe

procedure ADDCLSAN(S : motif)
    return ADDCLSCN(S) ;
end procedure

Algorithm 17 Génération de candidats par les opérations de suffixe pour la spécialisation de classe

```
1: procedure SPLCLSAN(S : motif)
2:     k ← |S.ς| ;
3:     F_temp ← ∅ ;
4:     if ∀i ∈ ℕ : 0 < i ≤ k, ∄ r(i,k) ∈ S.θ) then
5:         for all c ∈ C_Ω tel que c ⊑^r_Ω S.ς[k] do
6:             if (∀ A ∈ Ψ^n_c(c,S) : A < S.ς[k]) then
7:                 S' ← S ;
8:                 S'.ς[k] ← c;
9:                 F_temp ← F_temp ∪ {S'} ;
10:            end if
11:        end for
12:    end if
13:    return F_temp;
14: end procedure
```

Algorithm 18 Génération de candidats par les opérations de suffixe d'ajout de relation

```
1: procedure ADDRELAN(S : motif)
2:     k ← |S.ς| ;
3:     F_temp ← ∅ ;
4:     for all r ∈ R_Ω et P_r(r) = 1 tel que S.ς[k] ≪ ran(r) do
5:         for all i ∈ ℕ : 0 < i ≤ k tel que S.ς[k] ≪ ran(r,S.ς[i]) do
6:             if r(i,k) ∉ S.θ then
7:                 if (∀ r''(j1,j2) ∈ Ψ^1_r(r(i,k),S) : r''(j1,j2) < r(i,k)) then
8:                     F_temp ← F_temp ∪ {(S.ς, S.θ ∪ {r(i,k)})} ;
9:                 end if
10:            end if
11:        end for
12:    end for
13:    return F_temp;
14: end procedure
```

Algorithm 19 Génération de candidats par les opérations de suffixe pour la spécialisation de relation

```
1: procedure SPLRELAN(S : motif)
2:     k ← |S.ς| ;
3:     F_temp ← ∅ ;
4:     for all r(i,k) ∈ S.θ do
5:         for all r' ∈ R_Ω tel que r' ⊑^r_Ω r do
6:             if (∀ r''(j1,j2) ∈ Ψ^n_r(r'(i,k),S) − {r'(i,k)} : r''(j1,j2) < r(i,k)) then
7:                 F_temp ← F_temp ∪ {(S.ς, (S.θ − {r(i,k)}) ∪ {r'(i,k)})} ;
8:             end if
9:         end for
10:    end for
11:    return F_temp;
12: end procedure
```

5.3 Conclusion

Le présent chapitre vient de compléter la méthodologie de fouille de motifs par deux solutions algorithmiques qui s'inspirent du principe général de l'algorithme *Apriori* en adoptant une exploration progressive de l'espace des motifs et par élagage des motifs qui sont potentiellement non fréquents. Dans ce chapitre, nous avons successivement raffiné les opérations de spécialisation de motifs présentées dans le chapitre 4 en opérations élémentaires et par la suite en opérations canoniques. Les opérations élémentaires nous assurent un déplacement par niveau dans l'espace des motifs. Cependant, des générations multiples de motifs d'un même niveau sont possibles. Afin de réduire la redondance dans la génération des motifs, nous avons produit un ensemble d'opérations (opérations canoniques) où la spécialisation élémentaire de concept et de relation n'est autorisée que si le dernier concept du motif en question est impliqué. En se basant sur les opérations canoniques, nous avons développé l'algorithme *xPMiner* qui recherche tous les motifs fréquents.

Par la suite nous avons amélioré le processus de génération des motifs en éliminant complètement la génération dupliquée de motifs. Afin de réaliser cet objectif, nous avons introduit de nouvelles notions, tel que l'ensemble critique et de nouvelles relations pour comparer les différents éléments d'un motif. Finalement, nous avons développé et

implémenté un ensemble de quatre opérations de suffixe complet qui assurent une génération unique de chaque motif fréquent. L'algorithme résultant et appelé *xPMiner2* et sa complexité algorithmique est présentée dans l'annexe II.

Il reste à signaler que la formalisation du domaine de la fouille des motifs fréquents, a grandement facilité la tâche de l'extension du cadre initial par les opérations de suffixe.

Chapitre 6

Motifs de concepts/relations pour la recommandation Web

Dans le chapitre 1, nous avons fait le tour d'horizon des approches les plus utilisées dans le cadre de la personnalisation Web. Nous avons pu constater que ces approches ne profitent pas au complet de l'éventuelle présence de connaissances du domaine structurées. En effet, les connaissances utilisées souvent se limitent aux simples identifiants des objets et dans les meilleurs des cas on trouve des approches qui exploitent les taxonomies des catégories du domaine.

Dans le chapitre 4, nous avons présenté une approche pour intégrer les connaissances du domaine exprimées par une ontologie dans le processus de fouille de motifs. Ainsi, nous avons présenté une nouvelle catégorie de motifs construits autour des quatre éléments de base d'une ontologie, à savoir les individus (objets), les liens entre individus, les concepts et les rôles.

Dans le présent chapitre, nous présentons une stratégie de recommandation basée sur les motifs de concepts/relations.

6.1 Introduction

Un état de l'art des approches de recommandation existantes est présenté par G. Adomavicius et A. Tuzhilin dans [8]. Suivant le degré d'intégration des connaissances du domaine dans le processus, les systèmes de recommandation peuvent être classifiés en deux grandes catégories. Les systèmes de recommandation orientés item (objet) et les systèmes de recommandation orientés concept (catégorie d'objets). Dans les systèmes orientés objet, les objets fréquemment visités ensemble par un utilisateur ou

par un groupe d'utilisateurs sont recommandés lorsqu'au moins un de ces objets est visité par un nouvel utilisateur. Pour ce qui est des systèmes de recommandation orientés concept, les fréquences d'accès aux objets du contenu sont interprétées en fréquences d'accès aux différentes catégories de ces objets à différents niveaux d'abstraction. Ensuite, les instances des concepts apparaissant ensemble sont recommandées. Les motifs orientés concept sont semblables aux motifs de concepts/relations présentés dans le chapitre 4 à ceci près qu'ils ne couvrent pas les relations d'une ontologie. En effet, les motifs de concepts sont construits à partir de taxonomies de concepts. Par exemple, les motifs de concepts $\langle UrbanArea, Hotel\rangle$ et $\langle UrbanArea, Sightseeing\rangle$ correspondent respectivement aux motifs de concepts/relations $(\langle UrbanArea, Hotel\rangle, \{hasHotel(1,2)\})$, et $(\langle UrbanArea, Sightseeing\rangle, \{hasSightseeing(1,2)\})$ auxquels sont enlevées les relations.

Cependant, utiliser des motifs qui n'intègrent qu'une partie des connaissances du domaine constitue une limitation des systèmes de recommandation sous-jacents (voir la section 6.2).

Afin de combler les lacunes des systèmes de recommandation (orientés item ou concept), nous proposons une nouvelle stratégie de recommandation qui exploite les motifs et les opérateurs sur ces motifs présentés dans les chapitres 4 et 5. Principalement, il s'agit de développer une méthodologie de recommandation qui est à la fois précise, flexible et efficace [1, 3–5]. Pour cela, on commence par rechercher un ensemble de motifs de concepts/relations qui reflètent le comportement de l'utilisateur durant la session courante, et de recommander des objets qui instancient les classes de ces motifs, tout en ayant les relations exigées par le motif.

6.2 Recommandation orientée concept versus recommandation orientée objet

Les systèmes de recommandation orientés concept offrent une meilleure flexibilité que les systèmes de recommandation orientés objet. En effet, ces derniers se contentent

d'une représentation à un seul niveau d'abstraction du comportement des utilisateurs (niveau objet), alors que les systèmes orientés concept se basent sur les motifs de concepts qui permettent une capture à plusieurs niveaux. Cependant, les approches orientées concept souffrent d'un manque de précision.

De manière intuitive, la précision de la recommandation d'un objet à partir d'un motif est la capacité d'un système de recommandation à suggérer des objets qui constituent une suite "logique" compte tenu du motif et des objets déjà visités/consultés.

Pour illustrer les faiblesses des deux catégories d'approches, nous rappelons l'exemple déjà exposé dans le chapitre 1. Le scénario est comme suit : nous considérons un ensemble de trois utilisateurs tel que chacun d'eux ait visité une ville et un musée. Les paires des objets concernés sont respectivement : $\langle Montreal, The\_Montreal\_Museum\_of\_Fine\_Arts \rangle$, $\langle Paris, Louvre \rangle$, $\langle London, British\_Museum \rangle$. Maintenant, supposons un quatrième utilisateur qui a juste visité la ville *New_York*. La question est de retrouver parmi tous les objets ceux qui ont plus de chance d'être pertinents pour cet utilisateur en considérant le comportement de navigation des trois premiers utilisateurs. Nous observons que les trois premiers utilisateurs ont exploré les musées qui sont situés dans les villes qui ont été consultées auparavant. Intuitivement, le quatrième utilisateur se verra recommander de visiter les musées qui se trouvent dans la ville de *New_York*, tels que *American_Folk_Art*, *American_Numismatic_Society*, et *Bronx_Museum_of_the_Arts*. Cependant, de telles suggestions et un tel raisonnement n'est pas évident ni pour les systèmes de recommandation orientés item, ni pour ceux orientés concept. En effet, pour les systèmes orientés item, l'objet *New_York* n'a pas encore été consulté par les trois premiers utilisateurs et de ce fait, aucune recommandation ne peut être faite pour le quatrième utilisateur. Dans le cas des systèmes orientés concept, le comportement des trois utilisateurs peut être résumé par le motif $\langle City, Museum \rangle$. Ce motif permet de suggérer toutes les instances du concept *Museum* dans l'ontologie *Travel* qu'elles soient situées dans la ville de *New_York* ou pas. Ceci est une recommandation peu précise et le nombre de musées risque d'être élevé.

6.3 Recommandation dirigée par les motifs de concepts/ relations

L'exemple présenté dans la section 6.2 montre les limites des systèmes de recommandation qui sont soit orientés item ou orientés concept du domaine. Afin de combler ces limites, on devrait considérer des motifs flexibles (bon niveau d'abstraction), qui en même temps, offrent une bonne précision.

Les motifs composés de concepts et de relations sont extraits en utilisant l'algorithme *xPMiner2* tel que montré dans le chapitre 4. Par exemple, dans le cas du problème présenté dans le scénario de la section 6.2, le comportement des utilisateurs sera représenté en utilisant le motif $(\langle City, Museum\rangle, \{hasMuseum(1,2)\})$, et pour l'utilisateur qui a visité l'objet *New_York* seulement les instances du concept *Museum* qui sont référencées par un lien *hasMuseum* à partir de l'objet *New_York* seront suggérées. Ainsi, seulement les musées se trouvant dans la ville *New_York* seront recommandés pour le nouvel utilisateur.

Le principe de base de l'approche de recommandation que nous proposons consiste à rechercher les motifs de concepts/relations dont un des préfixes couvre la séquence d'objets que l'utilisateur courant a déjà consultés, et de lui suggérer des instances à partir des suffixes des motifs sélectionnés (voir définition 6.3.2). Il est à noter qu'un motif peut avoir plusieurs appariements avec la séquence d'objets. Pour une meilleure précision, nous avons choisi de considérer l'appariement correspondant au préfixe le plus long. Dans le cas où plusieurs appariements sont associés à ce préfixe, nous choisissons l'appariement dont l'image du dernier concept de ce préfixe est à la position la plus éloignée dans la séquence d'objets. Ainsi, on favorise la couverture des derniers objets consultés. De plus, les motifs sont triés par ordre descendant du nombre de relations qui partent du préfixe au suffixe d'un même motif. Ainsi, si on devrait recommander un maximum de m objets à la fois, on sélectionne les m instances à partir des premiers motifs de la liste ordonnée.

Les algorithmes nécessaires pour la mise en place de la méthodologie présentée ci-dessus sont détaillés dans le reste de cette section. L'algorithme principal, appelé

SemRAr (voir l'algorithme 20), permet de calculer la liste des objets à recommander. En premier lieu, l'algorithme transforme la séquence d'objets représentant la session courante de l'utilisateur en une séquence étendue s (voir ligne 8). Par la suite, *SemRAr* recherche les motifs qui couvrent partiellement s (voir les lignes 11 à 17) en faisant appel à l'algorithme 22. Pour chaque motif, les images du préfixe ainsi que l'indice du début du suffixe par rapport à s sont déterminés à l'aide de l'algorithme 20 (voir la ligne 12). Ensuite, ces données sont localement sauvegardées dans la table de hachage *ImagesCollection*. La clé de cette table correspond au nombre de relations qui partent du préfixe au suffixe d'un motif (voir la ligne 14). Finalement, la procédure GETOBJECTS, décrite dans l'algorithme 21, est appelée et un ensemble d'une cardinalité maximale m est retourné.

Algorithm 20 SemRAr

1: **Entrée :**
2: $\Omega, \mathscr{F}_\sigma$; ▷ Ontologie du domaine, motifs fréquents
3: O ; ▷ Liste des derniers objets visités
4: m ; ▷ Nombre maximum d'objets à recommander à la fois

5: **Sortie :**
6: $objSet$; ▷ Ensemble d'objets à suggérer

7: **Initialisation :**
8: $s \leftarrow$ ISEQTRANS(O) ;
9: $ImagesCollection \leftarrow \emptyset$; ▷ Initialisation de la collection d'images

10: **Méthode :**

11: **for all** $S \in \mathscr{F}_\sigma$ **do**
12: $i, Images \leftarrow$ COVERS(S, s) ;
13: **if** $i < |S.\varsigma|$ **then**
14: $p \leftarrow |\{r(j,k) \in S.\theta | j \leq i, k > i\}|$;
15: $ImagesCollection(p).add(S, i, Images)$;
16: **end if**
17: **end for**
18: $ObjSet \leftarrow$ GETOBJECTS$(\Omega, ImagesCollection, O, m)$;
19: **return** $ObjSet$;

Algorithm 21 GETOBJECTS : procédure qui calcule l'ensemble des objets à recommander

```
1: procedure GETOBJECTS(Ω : l'ontologie, ImagesCollection : collection de motifs
   avec les images de mise en correspondance, O : séquence d'objets, m)
2:     objSet ← ∅;                                   ▷ ensemble vide d'objets
3:     for all pSet ∈ ImagesCollection do
4:         for each elem ∈ pSet do
5:             i ← elem.i;
6:             S ← elem.S;
7:             objSet ← objSet ∪ {o ∈ O_Ω | o ∈ [S.ς[i]]_Ω ∧ ∀r(j,i) ∈
   S.θ, (O.ς[Images[j]], o) ∈ [r]_Ω};
8:         end for
9:     end for
10:    return objSet[1..m] ;
11: end procedure
```

Dans la procédure GETOBJECTS, les éléments de la collection des images sont traités séparément au sein de la boucle décrite par les lignes allant de 4 à 8. Pour chaque élément, on commence par récupérer la position i du début du suffixe, et le motif S de l'élément courant (lignes 5 et 6). Par la suite, les instances du concept se trouvant à cette position dans S sont filtrées comme suit. L'instance o de $S.\varsigma[i]$ n'est pas gardée si elle ne permet pas d'assurer le prolongement du préfixe de S. Autrement dit, pour que o soit gardée, il faudrait qu'elle supporte toutes les relations de $O.\theta$ qui aboutissent à la position i (ligne 7).

6.3.1 Couverture partielle d'une séquence d'objets par un motif

Un motif S couvre partiellement une séquence d'objets O, si et seulement si, la séquence d'objets étendue s, associée à O, est instance d'un préfixe de S (voir définition 6.3.1).

Définition 6.3.1. ***Relation de couverture partielle***

Soient S un motif de Γ_Ω, O une séquence d'objets de O^ω, et s la séquence d'objets étendue associée. On dit que S couvre O, dénoté par $S \rhd O$, s'il existe un préfixe de S qui

correspond homomorphiquement à une sous-structure de s. En clair, $S \rhd O$ s'il existe un motif $S' \in \Gamma_\Omega, s \lhd S'$ où :

- $|S'.\varsigma| < |S.\varsigma|$;
- $\forall i \in [1..|S'.\varsigma|], S'.\varsigma[i] = S.\varsigma[i]$;
- $\forall r(i_1,i_2) \in \rho, r(i_1,i_2) \in S.\theta \Leftrightarrow r(i_1,i_2) \in S'.\theta$.

En se basant sur l'ontologie *Travel*, la séquence d'objets $O_1 = \langle Surfline, Paris, Louvre \rangle$ est couverte par le motif $S_1 = (\langle Capital, Museum, LuxuryHotel \rangle, \{hasMuseum(1,2), hasLuxuryHotel(1,3)\})$ puisque le motif $S' = (\langle Capital, Museum \rangle, \{hasMuseum(1,2)\})$ qui est préfixe de S_1 résume O_1.

Il est à noter que lorsque le préfixe est égal au motif considéré, tester la couverture d'un motif pour une séquence d'objets revient à tester l'instanciation de ce motif pour la séquence considérée. Ainsi, la relation d'instanciation est considérée comme étant un cas particulier de la relation de couverture.

Maintenant que nous avons défini la notion de couverture, la notion de suffixe est définie comme suit.

Définition 6.3.2. ***Suffixe d'un motif***

Soient S un motif de Γ_Ω, O une séquence d'objets de O^ω, s la séquence d'objets étendue associée à O, avec $S \rhd O$, et S' un préfixe de S par rapport à O. Le suffixe de S par rapport à O et S' est une séquence de classes étendue S'' de Γ_Ω qui complète S' pour s'approcher le plus possible de S. Formellement :

- $S''.\varsigma = \langle S.\varsigma[i] \rangle_{|S'.\varsigma| < i \leq |S.\varsigma|}$;
- $S''.\theta = \{r(i,j) | r(i+|S'.\varsigma|, j+|S'.\varsigma|) \in S.\theta\}$.

Par exemple, le motif $(\langle Capital, LuxuryHotel \rangle, \{hasLuxuryHotel(1,2)\})$ est le suffixe du motif $S_1 = (\langle Capital, Museum, Capital, LuxuryHotel \rangle, \{hasMuseum(1,2), hasLuxuryHotel(3,4)\})$ par rapport au préfixe $(\langle Capital, Museum \rangle, \{hasMuseum(1,2)\})$ et à la séquence d'objets $O_1 = \langle Paris, Louvre, Roma, Grand\_Hotel\_Plaza \rangle$.

Comme, il est possible de retrouver plusieurs préfixes dont s est instance, la question suivante se pose : sur quel critère se baser pour sélectionner le préfixe ? Autrement dit, dans le cas où plusieurs appariements partiels sont possibles, lequel choisir ?

Une solution serait de favoriser les préfixes les plus longs. En effet, un long préfixe couvre plus d'objets qu'un préfixe plus court. Comme chaque appariement est caractérisé par la position du dernier concept de S qui admet une image dans s, nous privilégions la position la plus éloignée. C'est ainsi que l'appariement où le dernier concept est à la position la plus grande est sélectionné.

D'autres alternatives de choix d'un appariement existent. À titre d'exemple, il est possible de choisir un des appariements où les derniers objets consultés sont couverts.

Afin de tester la couverture d'un motif pour une séquence d'objets étendue et retrouver le préfixe à exploiter lors de la recommandation, l'algorithme COVERS est proposé (voir l'algorithme 22). Cet algorithme est très similaire à l'algorithme 3 de test de l'instanciation présenté dans le chapitre 4. La seule différence remarquable est qu'au lieu de rechercher des images pour tous les concepts et toutes les relations d'un motif, le nouvel algorithme recherche une mise en correspondance entre un préfixe du motif et la séquence d'objets étendue.

De plus, l'algorithme de calcul de la couverture exporte les *images* de la mise en correspondance entre s et S, ainsi que la position de la première classe de S qui vient juste après la classe qui est l'image du dernier objet de s. Ces images sont nécessaires pour le calcul de l'ensemble des objets à suggérer.

L'algorithme 22 recherche l'appariement maximal. À cette fin, on sauvegarde dans les variables *posIMax*, et *imgsPot* la plus grande position de tous les appariements déjà testés ainsi que les images associées à cet appariement. Lorsqu'on remet en cause un appariement, on vérifie si l'indice du dernier concept est supérieur à *posIMax*. Si oui, alors cet indice devient la nouvelle valeur de *posIMax*, et *imgsPot* est mis à jour avec les images de l'appariement en cours (lignes 14 à 17). L'algorithme de couverture se termine soit une fois qu'un appariement complet est retrouvé, soit que tous les appariements possibles ont été testés. Dans les deux cas, la position de la dernière classe de l'appariement maximal (s'il existe) est retournée conjointement avec les images associées.

Algorithm 22 Procédure de calcul de la couverture d'un motif

```
1: procedure COVERS(S : motif ; s : séquence d'objets étendue)
2:     posIMax ← 0 ;
3:     imgsPot ;          ▷ Tableau de listes contenant les images potentielles de chaque
   concept
4:     if |S.ς| > |s.ς| or |S.θ| > |s.θ| then
5:         return (posIMax, Images);
6:     end if
7:     imgsPot ← TROUVERIMAGESPOTENTIELLES(S, s);
8:     i ← 1;
9:     while i ≤ |S.ς| do
10:        if imgsPot[i].next? == false then
11:            if i == 1 then
12:                return (posIMax, Images);    ▷ Tous les appariements possibles ont
   été testés
13:            else
14:                if i > posIMax then
15:                    posIMax ← i ;
16:                    Images ← imgsPot.getCurrentElements[1..posIMax − 1] ;
17:                end if
18:                imgsPot[i].beforeFirst ;
19:                i − − ;
20:                break ;
21:            end if
22:        end if
23:        posImagCrt ← imgsPot[i].next() ;
24:        if i > 1 and posImagCrt < imgsPot[i − 1].current() then
25:            break ;
26:        end if
27:        if RELATIONSCORRESPONDENT(S, s, i, imgsPot, posImagCrt) then
28:            i + + ;
29:        end if
30:    end while
31:    if i == |S.ς| + 1 then
32:        Images ← imgsPot.getCurrentElements[1..|S.ς|];
33:        posIMax ← |S.ς|;
34:    end if
35:    return (posIMax, Images);
36: end procedure
```

6.4 Exemple d'application

Dans cette section, nous présentons un exemple de recommandation d'objets à partir des motifs découverts par l'algorithme *xPMiner2*.

6.4.1 Motifs générés par *xPMiner2*

Étant donné l'indisponibilité de jeux de données de sites Web où une ontologie est utilisée pour représenter les connaissances du domaine, nous utilisons un ensemble de données synthétiques composé de séquences d'objets. Les objets en question sont puisés d'une extension de l'ontologie *Travel* présentée dans le chapitre 3. Le tableau 6.1 présente un ensemble de onze séquences d'objets, et le tableau 6.2 montre la mise en correspondance entre les objets présents dans les séquences et leurs classes d'appartenance dans *Travel*. En outre, le support minimal (absolu) est fixé à 6 ($minsup = 6$).

Id	Séquence d'objets
1	$\langle France, Grenoble, Paris, Louvre \rangle$
2	$\langle Morocco, Rabat, Sofitel\_Diwan\_Rabat \rangle$
3	$\langle Coonabarabran, Warrumbungle\_National\_Park, Gumin\_Gumin\_Homestead \rangle$
4	$\langle Australia, Cairns, Clifton\_Beach, Cape\_York\_Safari, Montra\_Trilogy \rangle$
5	$\langle Canada, Forillon, Marche\_2Tiers, The\_2Campers \rangle$
6	$\langle Italy, Roma, Mausoleum\_of\_Augustus, Grand\_Hotel\_Plaza \rangle$
7	$\langle Bulgaria, Sophia, The\_National\_Gallery, Hotel\_Angel \rangle$
8	$\langle Algeria, Tizi, Djurdjura, Hotel\_El\_Arz, Tala\_Guilef \rangle$
9	$\langle Paris, Rodin, Royal\_Garden \rangle$
10	$\langle Cairns, Gazelle\_Tanzania\_Safari, Cifton\_Beach \rangle$
11	$\langle Teritoires\_Nord\_Ouest, Wood\_Buffalo, Little\_Buffalo \rangle$

Tableau 6.1 – Séquences formées d'objets appartenant à l'ontologie *Travel*.

Dans la première étape, les motifs candidats ont une classe seulement :
$S_1 = (\langle Accommodation \rangle, \{\}), S_2 = \langle \langle AccommodationRating \rangle, \{\} \rangle, S_3 = \langle \langle Activity \rangle, \{\} \rangle$, $S_4 = \langle \langle Contact \rangle, \{\} \rangle, S_5 = \langle \langle Destination \rangle, \{\} \rangle$. Par la suite, la procédure de test de can-

Classe	Instances (objets)
Country	*Australia, Algeria, Bulgaria, Canada, Italy, Morocco*
Capital	*Paris, Rabat, Roma, Sophia*
City	*Grenoble, Cairns*
Town	*Coonabarabran, Tizi*
Museum	*Louvre, Mausoleum_of_Augustus, Rodin, The_National_Gallery*
Hotel	*Montra_Trilogy, Hotel_El_Arz*
LuxuryHotel	*Sofitel_Diwan_Rabat, Grand_Hotel_Plaza, Royal_Garden, Hotel$_A$ngel*
NationalPark	*Warrumbungle_National_Park, Forillon, Wood_Buffalo, Djurjura*
Campground	*GUMIN_GUMIN_HOMESTEAD, Little _Buffalo, The_2Campers*
Beach	*Clifton_Beach*
Safari	*Gazelle_Tanzania_Safari, Cape_York _Safari*
Hiking	*Marche_2Tiers*
Skiing	*Tala_Guilef*

Tableau 6.2 – Mise en correspondance classe/instance.

didats (voir l'algorithme 3 du chapitre 4) calcule pour chaque motif candidat le nombre de séquences d'objets qui l'instancient. Par exemple, S_1 échoue le test avec la première et la dixième séquences, mais passe avec succès le test d'instanciation avec le reste des séquences. Le support de chacun des cinq motifs est présenté dans le tableau 6.3. De ce tableau, nous constatons que S_2 et S_4 ont un support égal à 0. En effet, il est facile à vérifier que pour toutes les séquences d'objets, il n'existe aucun objet qui peut être associé avec la classe *AccommodationRating* de S_2. Il est de même pour le concept *Contact* de S_4.

À ce niveau, la propriété d'anti-monotonie de la fréquence garantit que tous les motifs qui peuvent être générés à partir d'un motif non fréquent sont à leur tour non fréquents. Le reste des candidats ont une fréquence supérieure au seuil *minsup* fixé à 6. Ainsi, S_1, S_3, et S_5 sont marqués comme étant fréquents et seront considérés comme une entrée pour l'itération suivante.

Motif	Support
S_1	9
S_2	0
S_3	8
S_4	0
S_5	11

Tableau 6.3 – Motifs de rang 1 et leurs supports.

Chaque motif *1-fréquent* est étendu par le moyen des opérations de suffixe, donnant lieu à l'ensemble des *2-candidats*. En considérant que dans l'ontologie *Travel* le concept *Destination* se spécialise en huit concepts, alors l'ensemble des *2-candidats* pour S_5 est le suivant : $(\langle BackpakerDestination\rangle, \{\})$, $\langle\langle Beach\rangle, \{\})$, $\langle\langle BudgetHotelDestination\rangle$, $\{\})$, $(\langle FamilyDestination\rangle, \{\})$, $(\langle QuietDestination\rangle, \{\})$, $(\langle RetireeDestination\rangle, \{\})$, $(\langle RuralArea\rangle, \{\})$ et $(\langle UrbanArea\rangle, \{\})$.
De manière identique, l'insertion de concept à S_5 aboutit aux *2-candidats* suivants : $(\langle Destination, Accommodation\rangle, \{\})$, $(\langle Destination, AccommodationRating\rangle, \{\})$, $(\langle Destination, Contact\rangle, \{\})$ et $(\langle Destination, Activity\rangle, \{\})$. La liste de tous les *2-candidats* est présentée dans le tableau 6.4.

Motif	Support
$(\langle BackpakerDestination\rangle, \{\})$	0
$(\langle Beach\rangle, \{\})$	2
$(\langle BudgetHotelDestination\rangle, \{\})$	0
$(\langle FamilyDestination\rangle, \{\})$	0
$(\langle QuietDestination\rangle, \{\})$	0
$(\langle RetireeDestination\rangle, \{\})$	0
$(\langle RuralArea\rangle, \{\})$	4
$(\langle UrbanArea\rangle, \{\})$	9
$(\langle Destination, Accommodation\rangle, \{\})$	9
$(\langle Destination, AccommodationRating\rangle, \{\})$	0
$(\langle Destination, Contact\rangle, \{\})$	0
$(\langle Destination, Activity\rangle, \{\})$	8

Tableau 6.4 – Candidats de rang 2 (*2-candidats*) avec leurs supports

Il à noter que les opérations qui impliquent des relations ne sont applicables qu'à

partir de la deuxième itération.

Maintenant, les *2-candidats* sont filtrés par le test d'instanciation pour déterminer les motifs *2-fréquents*. Nous observons que seulement les motifs ($\langle Destination, Accommodation\rangle, \{\}$), ($\langle UrbanArea\rangle, \{\}$) et ($\langle Destination, Activity\rangle, \{\}$) sont fréquents (voir le tableau 6.5). Ces motifs, ayant un rang de 2, sont ensuite utilisés pour générer les motifs *3-fréquents*.

Motif	Support
($\langle UrbanArea\rangle, \{\}$)	9
($\langle Destination, Accommodation\rangle, \{\}$)	9
($\langle Destination, Activity\rangle, \{\}$)	8

Tableau 6.5 – Motifs fréquents de rang 2 et support associé (*2-fréquents*)

Si nous considérons l'extension du motif ($\langle Destination, Activity\rangle, \{\}$), les candidats générés par la spécialisation de *Activity* sont : ($\langle Destination, Adventure\rangle, \{\}$), ($\langle Destination, Relaxation\rangle, \{\}$), ($\langle Destination, Sightseeing\rangle, \{\}$), ($\langle Destination, Sports\rangle, \{\}$), et ($\langle Activity, Accommodation\rangle, \{\}$). On remarque que plusieurs insertions de concept conduisent à la génération des motifs : ($\langle Destination, Activity, AccommodationRating\rangle, \{\}$), ($\langle Destination, Activity, Contact\rangle, \{\}$), et ($\langle Destination, Activity, Destination\rangle, \{\}$). Le motif ($\langle Destination, Activity\rangle, \{hasActivity(1,2)\}$) est le seul qui est obtenu par l'insertion de relation. Le tableau 6.6 fournit les fréquences des candidats générés dans la troisième itération.

Motif	Support
($\langle City\rangle, \{\}$)	8
($\langle Destination, Accommodation\rangle$, $\{hasAccommodation(1,2)\}$)	9
($\langle UrbanArea, Activity\rangle, \{\}$)	9
($\langle UrbanArea, Accommodation\rangle, \{\}$)	8
($\langle Destination, Activity\rangle, \{hasActivity(1,2)\}$)	7
($\langle Destination, Activity, Accommodation\rangle, \{\}$)	6

Tableau 6.6 – Motifs fréquents de rang 3 (*3-fréquents*) et leurs supports.

À la quatrième itération, et après la génération et le test de l'ensemble des *4-candidats*, nous obtenons les motifs *4-fréquents* présentés dans le tableau 6.7.

Motif	Support
$(\langle City, Activity\rangle, \{\})$	6
$(\langle UrbanArea, Accommodation\rangle, \{hasAccommodation(1,2)\})$	8
$(\langle UrbanArea, Hotel\rangle, \{\})$	6
$(\langle UrbanArea, Sightseeing\rangle, \{\})$	6
$(\langle UrbanArea, Activity\rangle, \{hasActivity(1,2)\})$	7
$(\langle Destination, Hotel\rangle, \{hasAccommodation(1,2)\})$	6
$(\langle Destination, Activity, Accommodation\rangle, \{hasAccommodation(1,3)\})$	6
$(\langle Destination, Activity, Accommodation\rangle, \{hasActivity(1,2)\})$	6

Tableau 6.7 – Motifs fréquent de rang 4 (*4-fréquents*) et leurs supports.

Les tableaux 6.8 et 6.9 présentent respectivement les motifs fréquents de rang 5 et 6. Le tableau 6.10, quant à lui, présente la liste des *7-candidats* avec leurs supports.

Motif	Support
$(\langle UrbanArea, Hotel\rangle, \{hasAccommodation(1,2)\})$	6
$(\langle UrbanArea, Sightseeing\rangle, \{hasActivity(1,2)\})$	6
$(\langle Destination, Hotel\rangle, \{hasHotel(1,2)\})$	6
$(\langle Destination, Activity, Accommodation\rangle, \{hasActivity(1,2), hasAccommodation(1,3)\})$	6

Tableau 6.8 – Motifs fréquents de rang 5 (*5-fréquents*) et leurs supports.

Finalement, le processus de fouille s'arrête à la septième itération car aucun des *7-candidats* n'est fréquent.

Motif	Support
$(\langle UrbanArea, Hotel\rangle, \{hasHotel(1,2)\})$	6
$(\langle UrbanArea, Sightseeing\rangle, \{hasSightseeing(1,2)\})$	6

Tableau 6.9 – Motifs fréquents de rang 6 (*6-fréquents*) et leurs supports.

Motif	Support
$(\langle UrbanArea, LuxuryHotel\rangle, \{hasHotel(1,2)\})$	4
$(\langle UrbanArea, Hotel, AccommodationRating\rangle, \{hasHotel(1,2)\})$	0
$(\langle UrbanArea, Hotel, Contact\rangle, \{hasHotel(1,2)\})$	0
$(\langle UrbanArea, Hotel, Destination\rangle, \{hasHotel(1,2)\})$	0
$(\langle UrbanArea, Hotel, Accommodation\rangle, \{hasHotel(1,2)\})$	0
$(\langle UrbanArea, Hotel, Activity\rangle, \{hasHotel(1,2)\})$	1
$(\langle UrbanArea, Museum\rangle, \{hasSightseeing(1,2)\})$	4
$(\langle UrbanArea, Safari\rangle, \{hasSightseeing(1,2)\})$	1
$(\langle UrbanArea, Sightseeing, AccommodationRating\rangle, \{hasSightseeing(1,2)\})$	0
$(\langle UrbanArea, Sightseeing, Contact\rangle, \{hasSightseeing(1,2)\})$	0
$(\langle UrbanArea, Sightseeing, Destination\rangle, \{hasSightseeing(1,2)\})$	0
$(\langle UrbanArea, Sightseeing, Accommodation\rangle, \{hasSightseeing(1,2)\})$	1
$(\langle UrbanArea, Sightseeing, Activity\rangle, \{hasSightseeing(1,2)\})$	4

Tableau 6.10 – Motifs candidat de rang 7 (*7-candidats*) et leurs supports.

6.4.2 Recommandation d'objets

Dans cette section, nous soulignons l'avantage de l'utilisation des motifs découverts précédemment pour la suggestion/recommandation d'objets.

Dans la suite, nous illustrons les trois différentes approches de recommandation qui se basent respectivement sur les *objets*, les *concepts*, et celle qui se base sur les *concepts et relations*, et nous soulignons l'avantage de l'utilisation des motifs découverts précédemment pour la suggestion/recommandation d'objets. Dans la première approche, les items (objets) qui sont fréquemment consultés ensemble par plusieurs utilisateurs sont recommandés lorsqu'au moins un de ces objets est sélectionné par un nouvel utilisateur. De l'ensemble des séquences d'objets du tableau 6.1, les seuls objets qui apparaissent plus qu'une fois, sont *Paris* et *Cairns* avec une fréquence inférieure au seuil *minsup*. Dans l'approche orientée concepts, les fréquences d'accès sont calculées sur

les concepts du domaine à différents niveaux d'abstraction. Par la suite, les instances des concepts qui apparaissent ensemble sont recommandées. Les motifs composés de concepts sont identiques à ceux qui sont retrouvés par *xPMiner2*, excepté qu'ils ne couvrent pas les relations de l'ontologie. Par exemple, les motifs de concepts/relations $(\langle UrbanArea, Hotel\rangle, \{hasHotel(1,2)\})$, et $(\langle UrbanArea, Sightseeing\rangle, \{hasSightseeing(1,2)\})$ correspondent respectivement aux motifs de concepts purs (qui seront également retrouvés par *xPMiner2*) $\langle UrbanArea, Hotel\rangle$ et $\langle UrbanArea, Sightseeing\rangle$.

Derniers objets visités	Recommandations		
	Approches classiques		Concepts/relations
	basée sur les items	basée sur les concepts	xPMiner2
Montreal	{}	{Toutes les instances directes et indirectes des classes *Hotel* et *Sightseeing*, de l'ontologie *Travel*}	{The_Montreal_Museum _of_Fine_Arts, Pointe_à_Callière} {Le Saint_Malo, Pierre _du_Calvet}
Central_Florida	{}	{Toutes les instances directes et indirectes des classes *Activity* et *Accommodation* de l'ontologie *Travel*}	{The_Westin_Diplomat _Resort_and_Spa} {Surfline}

Tableau 6.11 – Objets suggérés suivant différentes stratégies de recommandation.

Étant donné le rôle essentiel de la co-occurrence des objets dans la stratégie de recommandation orientée objets, un objet ne sera pas suggéré si la fréquence de ses apparitions n'atteint pas un certain seuil. Par exemple, dans le tableau 6.11, il n'existe aucun objet à recommander pour un utilisateur ayant consulté les objets *Montreal* et *Central_Florida*. En effet, ces objets n'apparaissent pas ensemble dans les séquences de données initiales. Par contre, une stratégie de recommandation orientée concepts retrouvera des objets à recommander en se basant sur les motifs de concepts dérivés des motifs de concepts/relations des tableaux 6.8 et 6.9. Ces motifs sont : $\langle UrbanArea, Hotel\rangle$, $\langle UrbanArea, Sightseeing\rangle$, $\langle Destination, Hotel\rangle$, et $\langle Destination, Activity, Accommodation\rangle$. Ainsi, les instances des concepts *Hotel* et *Sightseeing* pourraient être

recommandées.

Il est à noter qu'avec cette façon de procéder, aucune distinction n'est faite entre les hôtels situés dans Montréal et ceux situés dans d'autres villes. La même remarque reste valable pour le concept *Sightseeing* dont les instances appartiennent soit à *Safari* soit à *Museum*, et seront recommandées indépendamment du fait qu'elles soient situées dans Montréal ou pas. Ainsi, la qualité des recommandations pourrait être affectée.

Avec notre approche d'intégration des concepts et des relations dans les motifs, nous faisons des recommendations à la fois cohérentes et non triviales pour les approches existantes grâce à la prise en considération de l'ontologie du domaine (liens inter-objets et relations inter-concepts). Par exemple, pour assister les utilisateurs qui ont sélectionné *Montreal*, l'approche que nous proposons se base sur les motifs ($\langle UrbanArea, Hotel\rangle$, $\{hasHotel(1,2)\}$) et ($\langle UrbanArea$, $Sightseeing\rangle$, $\{hasSightseeing(1,2)\}$) pour suggérer les objets *The_Montreal_Museum_of_Fine_Arts* et *Pointe_à_Callière* qui sont instances de *Museum*, et les objets *Le_Saint_Malo* et *Pierre_du_Calvet* qui sont instances de *Hotel*. Chacune de ces instances est connectée à l'objet *Montreal* dans l'ontologie *Travel* via un lien relationnel.

Pour les utilisateurs qui ont sélectionné *Florida_Beach*, aucun motif *6-fréquent* (voir le tableau 6.9) ne correspond à leur choix. Cependant, en considérant les motifs du tableau 6.8, le motif ($\langle Destination, Activity, Accommodation\rangle$, $\{hasActivity(1,2)$, $hasAccommodation(1,3)\}$) servira pour la suggestion d'une instance de *Surf* — sous-classe de *Activity* — qui est associée à *Florida_beach* avec le lien *hasSurf*. Par contre l'autre motif, ($\langle Destination$, $Hotel\rangle, \{hasHotel(1,2)\}$), ne permet de générer aucune suggestion.

6.5 Conclusion

Nous venons de montrer comment appliquer les motifs de concepts/relations pour combler les lacunes des approches de recommandation qui n'exploitent pas toutes les connaissances ontologiques du domaine. L'approche de recommandation proposée a l'originalité d'être basée sur une double procédure de filtrage. Au premier niveau de

filtrage, on calcule un ensemble de motifs résumant au mieux le comportement courant de navigation de l'utilisateur. Avec le deuxième niveau de filtrage, un maximum de m instances sont retournées à l'utilisateur.

Dans ce chapitre, nous avons pu constater le potentiel des motifs de concepts/relations pour la recommandation Web. En effet, l'inclusion des concepts et des relations dans le processus de fouille nous permet d'avoir une bonne précision tout en raisonnant à des niveaux d'abstraction variables.

Chapitre 7

Implémentation et expérimentations

Aux chapitres 4 et 5, nous avons présenté deux langages, l'un pour la description des données et l'autre pour la description des motifs composés de concepts et de relations puisés de l'ontologie du domaine, ainsi qu'une méthodologie de fouille de ce type de motifs. Rappelons que les motifs que nous recherchons constituent une nouvelle catégorie et les infrastructures et les outils existants ne nous permettent pas de les manipuler. Dans ce chapitre, nous présentons une plateforme pour la fouille des motifs de concepts/relations. Par la suite, nous exposons les résultats des expérimentations que nous avons effectuées sur l'implémentation de l'algorithme *xPMiner2* proposé dans le chapitre 5.

7.1 *OntoMiner* : plateforme pour la fouille des motifs de concepts/relations

Les ontologies jouent un rôle central dans le Web sémantique : elles offrent un modèle formel des connaissances d'un domaine pouvant être exploité pour les différentes tâches inhérentes du Web sémantique telles que l'inférence et le raisonnement. Les domaines d'application des ontologies sont très variés, allant de la recherche médicale et biologique, au commerce électronique et aux sciences sociales. Le développement d'outils spécifiques venant compléter la liste de ceux déjà existants ne fait que contribuer à l'élargissement et à l'avancement du Web sémantique. Ainsi donc, des solutions spécifiques doivent être apportées afin de tenir compte des difficultés et des particularités d'un domaine spécifique. La personnalisation Web qui se base sur les ontologies est une des applications liées au Web sémantique.

Le Web sémantique est doté d'outils de visualisation et de manipulation d'ontologies

tel que le système *Protégé* [79] qui est considéré comme l'une des plateformes les plus complètes [34].

Notre objectif est de construire un outil flexible pour l'extraction et la manipulation des motifs. Dans cette section, nous présentons *OntoMiner* une plateforme pour la personnalisation ayant les objectifs suivants : (i) implémenter les langages des données et des motifs, (ii) intégrer l'algorithme *xPMiner2*, et (iii) servir d'environnement de test et d'expérimentation.

7.1.1 Architecture d'*OntoMiner*

OntoMiner est l'outil que nous avons développé pour manipuler les données et générer les motifs composés de concepts/relations. L'architecture générale de ce système est présentée dans la figure 7.1. Le noyau d'*OntoMiner* est constitué du module de simulation de sessions utilisateurs, du module de transformation de séquences d'objets et du module de fouille de motifs. La figure 7.1 illustre les principaux composants de notre plateforme de fouille. Nous détaillons dans la suite chacun de ces composants.

7.1.1.1 Simulation de sessions

OntoMiner est doté d'un éditeur de séquences d'objets pour simuler les sessions des utilisateurs qui naviguent dans un site dont les objets sont décrits par une ontologie du domaine.

Ainsi, l'utilisateur d'*OntoMiner* a la possibilité de parcourir les objets de façon similaire à ce qui se fait dans les cas réels : parcours par catégorie, passage d'un objet à un autre en suivant les liens offerts. En plus de la fonctionnalité de navigation qui est offerte aux utilisateurs finaux, l'administrateur a la possibilité d'ajouter des objets à la base initiale en instanciant des concepts de l'ontologie du domaine.

Pour sauvegarder les séquences d'objets créées par un utilisateur pour une exploitation ultérieure, *OntoMiner* génère un fichier *XML* respectant le format décrit par les règles représentées dans la figure 7.2. De plus, *OntoMiner* offre la possibilité d'importer un ensemble de séquences d'objets à partir d'un fichier *XML* ayant le format décrit par

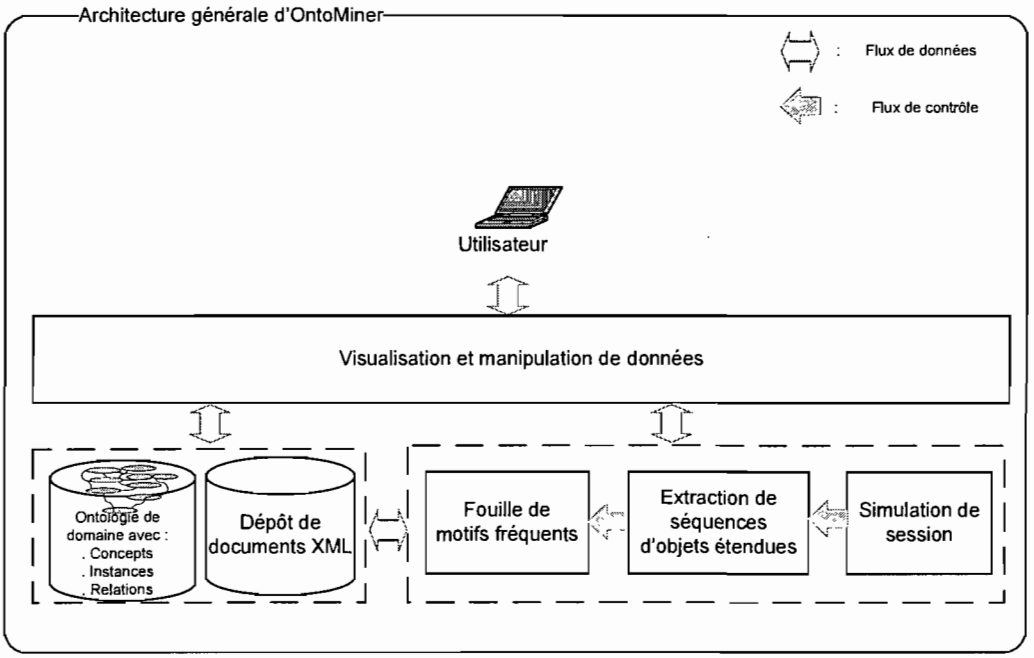


Figure 7.1 – Architecture générale d'*OntoMiner* : plateforme pour la génération et la manipulation de motifs.

les règles de la figure 7.2. L'ensemble des séquences d'objets est représenté par l'élément *Sequences* qui peut contenir plusieurs éléments de type *Sequence*. Chaque élément *Sequence* doit contenir au moins un objet représenté par l'élément *SeqObject* qui a l'attribut *name* dont la valeur représentera le nom de cet objet.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.ontominer.org/Sequences"
    xmlns:tns="http://www.ontominer.org/Sequences">
     <element name="Sequences" type="tns:Sequences">
     </element>

     <complexType name="Sequences">
         <sequence>
             <element minOccurs="0" maxOccurs="unbounded"
                 name="Sequence" type="tns:Sequence">
             </element>
         </sequence>
         <attribute name="name" type="string"></attribute>
     </complexType>
     <complexType name="Sequence">
         <sequence>
             <element name="SeqObject" minOccurs="1" maxOccurs="unbounded">
                 <complexType>
                     <attribute name="name" type="string"></attribute>
                 </complexType>
             </element>
         </sequence>
         <attribute name="name" type="string"></attribute>
     </complexType>
</schema>
```

Figure 7.2 – Schéma *XSD* décrivant la structure de documents XML pour représenter les séquences d'objets.

7.1.1.2 Transformation de séquences

Le module de transformation s'occupe d'enrichir les séquences simples qui deviennent des séquences étendues telles que définies dans le chapitre 4. En résumé, la tâche consiste à augmenter chaque séquence d'objets par les liens relationnels existants entre les objets de la séquence. Il est à noter que seulement les liens co-linéaires avec l'ordre des objets dans la séquence sont pris en considération, et que ces liens sont puisés dans l'ontologie du domaine. Les séquences d'objets étendues sont stockées sous format XML, en suivant la grammaire représentée dans la figure 7.3. L'ensemble des séquences d'objets étendues est représenté par l'élément *extobjseqs* qui peut contenir plusieurs sous-éléments. Une

séquence d'objets étendue est représentée par l'élément *extobjseq* et est composée d'un sous-élément *objects* pour représenter la liste des objets de la séquence et d'un élément *relations* pour représenter l'ensemble des relations de la séquence. Alors qu'un élément *object* est composé d'un seul sous-élément ayant l'attribut *name*, un élément *relation* est composé de deux autres attributs : *obj-in* et *obj-out* qui font respectivement référence aux positions de départ et d'arrivée de la relation en question dans la séquence considérée.

7.1.2 Exploration de l'espace des motifs

Le module de fouille de motifs prend en entrée un ensemble de séquences d'objets étendues et produit l'ensemble des motifs fréquents sous-jacents.

Ce module est conçu pour prendre en compte différents algorithmes. En effet, des interfaces génériques sont utilisées pour définir les méthodes qui sont appelées lors de la fouille de motifs fréquents. Ainsi, tout algorithme qui implémente ces interfaces est automatiquement intégré dans *OntoMiner*. Cela rend la plateforme plus flexible et adaptable à d'autres approches de fouille.

Actuellement, l'extraction des motifs est effectuée par l'algorithme *xPMiner2* présenté dans le chapitre 5.

Une fois que les motifs sont extraits, *OntoMiner* offre la possibilité de les exporter sous format *XML* en générant un document qui satisfait les règles *XSD* présentées dans la figure 7.5. Une interface graphique résumant les fonctionnalités de base de la plateforme *OntoMiner* est présentée dans la figure 7.4. On y distingue les éléments suivants. Une interface de visualisation et de manipulation de séquences d'objets qui est composée d'un éditeur de séquences (voir les éléments pointés par la flèche *A*) et d'une représentation sous forme d'une arborescence des séquences (voir la fenêtre pointée par la flèche *C*). La fenêtre pointée par la flèche *E* représente l'arborescence de séquences d'objets étendues. Le support minimal est saisi dans la zone d'édition pointée par *D* et les motifs générés sont affichés sous forme d'une arborescence dans la fenêtre pointée par la flèche *F*. Finalement, l'importation et l'exportation des séquences d'objets et des motifs est faite à travers les options pointées par la flèche *B*.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.ontominer.org/ExtObjSeqSchema"
    xmlns:tns="http://www.ontominer.org/ExtObjSeqSchema">
    <element name="extobjseqs" type="tns:extobjseqs"/>
    <complexType name="extobjseqs">
        <sequence>
            <element name="name" type="string"/>
            <element name="owl-source">
                <complexType>
                    <attribute name="project-name" type="string"/>
                    <attribute name="owl-url" type="anyURI"/>
                </complexType>
            </element>
            <element name="extobjseqs" type="tns:extobjseqs"/>
        </sequence>
        <attribute default="0" name="nbr-extobjseqs" type="integer"/>
    </complexType>
    <complexType name="extobjseqs">
        <sequence>
            <element minOccurs="0" maxOccurs="unbounded" name="extobjseq" type="tns:extobjseq"/>
        </sequence>
    </complexType>
    <complexType name="extobjseq">
        <sequence>
            <element minOccurs="0" maxOccurs="1" name="objects" type="tns:objects"/>
            <element minOccurs="0" maxOccurs="1" name="relations" type="tns:relations"/>
        </sequence>
        <attribute name="name" type="string" default="extobjseq"/>
    </complexType>
    <complexType name="objects">
        <sequence>
            <element minOccurs="0" maxOccurs="unbounded" name="object">
                <complexType> <attribute name="name" type="string"/></complexType>
            </element>
        </sequence>
    </complexType>
    <complexType name="relations">
        <sequence>
            <element minOccurs="0" maxOccurs="unbounded" name="relation">
                <complexType>
                    <attribute name="name" type="string"></attribute>
                    <attribute name="obj-in" type="int"></attribute>
                    <attribute name="obj-out" type="int"></attribute>
                </complexType>
            </element>
        </sequence>
    </complexType>
    <simpleType name="ID">
        <restriction base="string"> <pattern value="\d{5}"/></restriction>
    </simpleType>
</schema>
```

Figure 7.3 – Schéma *XSD* décrivant la structure de documents XML pour représenter les séquences d'objets étendues.

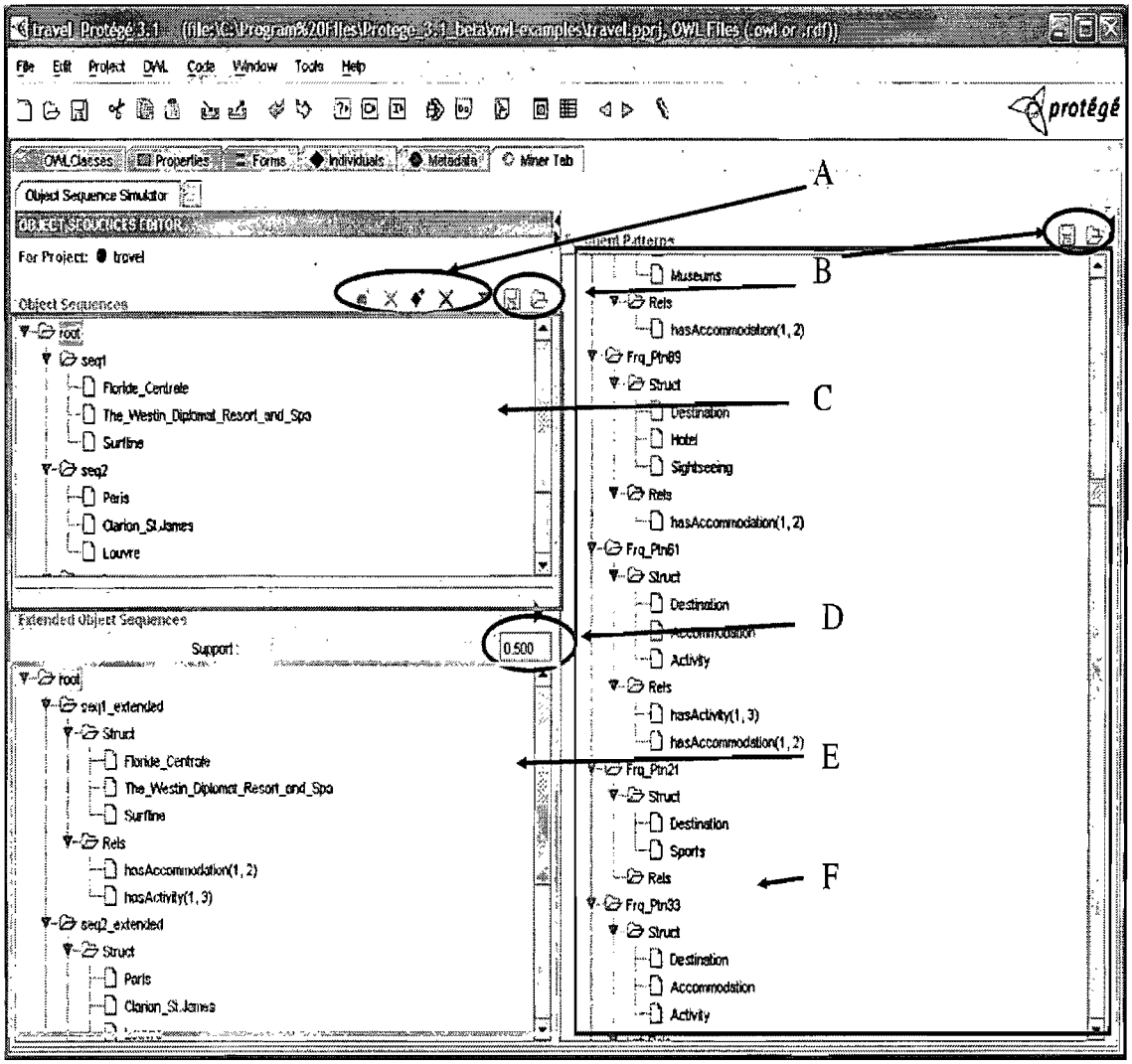


Figure 7.4 – Interface graphique d'*OntoMiner*.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.ontominer.org/PatternSchema"
    xmlns:trs="http://www.ontominer.org/PatternSchema">
    <element name="miner-patterns" type="tns:miner-patterns"/>
    <complexType name="miner-patterns">
        <sequence>
            <element name="name" type="string"/>
            <element name="owl-source">
                <complexType>
                    <attribute name="project-name" type="string"/>
                    <attribute name="owl-url" type="anyURI"/>
                </complexType>
            </element>
            <element name="patterns" type="tns:patterns"/>
        </sequence>
        <attribute default="0" name="non-patterns" type="integer"/>
    </complexType>
    <complexType name="patterns">
        <sequence>
            <element minOccurs="0" maxOccurs="unbounded" name="pattern" type="tns:pattern"/>
        </sequence>
        <attribute name="range" type="int"/>
    </complexType>
    <complexType name="pattern">
        <sequence>
            <element minOccurs="0" maxOccurs="1" name="concepts" type="trs.concepts"/>
            <element minOccurs="0" maxOccurs="1" name="relations" type="tns:relations"/>
        </sequence>
        <attribute name="name" type="string" default="pattern"/>
    </complexType>
    <complexType name="concepts">
        <sequence>
            <element minOccurs="0" maxOccurs="unbounded" name="concept">
                <complexType> <attribute name="name" type="string"/></complexType>
            </element>
        </sequence>
    </complexType>
    <complexType name="relations">
        <sequence>
            <element minOccurs="0" maxOccurs="unbounded" name="relation">
                <complexType>
                    <attribute name="name" type="string"></attribute>
                    <attribute name="opt-ca" type="int"></attribute>
                    <attribute name="opt-cat" type="int"></attribute>
                </complexType>
            </element>
        </sequence>
    </complexType>
    <simpleType name="ID">
        <restriction base="string"> <pattern value="\c{5}"/></restriction>
    </simpleType>
</schema>
```

Figure 7.5 – Schéma *XSD* décrivant la structure de documents XML pour représenter les motifs.

7.2 *OntoMiner* et le noyau de *Protégé*

Les noyaux de *Protégé* et du plugiciel qui prend en charge le language OWL ont une architecture ouverte et leur code source est disponible (*open source*). Afin de doter *OntoMiner* d'une flexibilité et d'une extensibilité pour des ajouts futurs, nous avons développé un ensemble de primitives de fouille et de manipulation de motifs de concepts / relations au dessus de l'API de *Protégé*. Des composants de ces noyaux sont exploités et offrent les avantages suivants : (i) bénéficier de l'expérience de *Protégé* liée à la manipulation d'ontologies, (ii) exploiter les capacités de raisonnement offertes, (iii) intégrer les services Web, (iv) faciliter le développement d'extensions pour d'autres types de motifs. Une vue d'ensemble de ce système est présentée dans la figure 7.6.

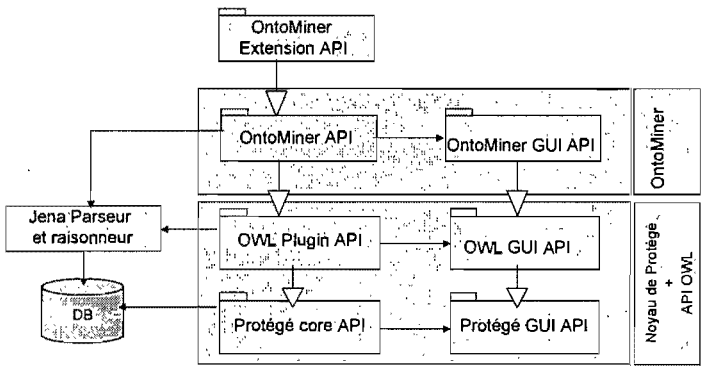


Figure 7.6 – *OntoMiner* et le noyau de Protégé.

7.3 Expérimentations

Dans cette section, nous rapportons les résultats des expérimentations sur la mise à l'échelle de l'algorithme *xPMiner2* lors de la génération de motifs à partir d'une base de séquences d'objets étendues. Toutes les expériences ont été menées sur un ensemble de machines virtuelles ayant la même configuration. Chaque machine dispose des ressources suivantes :

- Processeur *Intel* 32 *bits* avec une fréquence d'horloge de 2,33Ghz [1] ;
- Système d'exploitation est un *CentOS Server* 5 avec un noyau Linux 2.6.18 ;
- Mémoire principale de chaque machine est de 400 Méga octets, avec swap de 2 Giga octets ;
- *Java(TM) SE Runtime Environment* version 1.6.0_05.

En raison de la non disponibilité de données réelles qui sont décrites par une ontologie du domaine, les expérimentations ont été effectuées sur des données synthétiques. Ces données sont automatiquement générées par le plugiciel selon un processus bien défini (voir la section 7.3.2) et qui est inspiré des travaux de Agrawal et Srikant [10, 75]. Le modèle imite le comportement d'un utilisateur qui navigue à travers des objets de contenu qui sont supposés être visités/sélectionnés par les utilisateurs.

7.3.1 Source de données

La source de données que nous utilisons est l'ontologie *Travel* qui décrit le domaine de tourisme. Les objets sont décrits par l'ontologie *Travel* qui est composée de 36 classes, 44 relations et 99 objets distincts.

7.3.2 Génération de données synthétiques

Dans notre modèle de données, une séquence est composée d'une liste d'objets visités/achetés/sélectionnés par un utilisateur dans une application Web. La question qui se

1. Dans le cadre de cette thèse, la virgule représente le point décimal.

pose est de savoir comment faire pour générer des séquences d'objets qui simulent les sessions d'un utilisateur.

Dans notre tentative de répondre à cette question, nous proposons un modèle de génération automatique de séquences d'objets. Dans ce processus, le comportement de parcours des objets de contenu est imité. Dans le modèle que nous proposons, les utilisateurs ont tendance à visiter des objets d'un site qui sont liés. Cependant, les utilisateurs pourraient visiter des objets de contenu qui ne sont pas sémantiquement liés. Le processus de génération de jeux de données que nous avons développé prend en considération de telles possibilités.

Concrètement, le processus de génération est comme suit. Premièrement, nous sélectionnons un ensemble de concepts, appelé *réserve de concepts*, dont les instances serviront pour le choix du premier objet d'une séquence. Les instances de cet ensemble ont plus de chance d'être visitées comme premiers objets d'une séquence. Par exemple, peuvent faire partie de cette réserve, les catégories des objets des pages d'accueil des sites Web. Dans le cas de l'ontologie *Travel*, les instances de *Destination* (*countries, towns, national parks, etc.*) et *Activity* (*safaris, museums, sports, etc.*) sont susceptibles d'être sélectionnées en premier.

Deuxièmement, la taille d'une séquence n est tirée de l'intervalle [*minsize*, *maxsize*] en utilisant une distribution de *Poisson* [12] pour s'assurer que la taille de la séquence est complètement indépendante des tailles des séquences déjà générées.

Le premier objet de la séquence est sélectionné en deux étapes. Dans la première étape, un concept est sélectionné de la réserve. Dans la deuxième étape, une instance est tirée de l'extension du concept sélectionné. Dans les deux cas, une distribution *Gaussienne* [12] est appliquée.

Par la suite, une série de tirages est effectuée pour choisir les objets qui vont prendre les positions de 2 à n. Dans le premier tirage, une distribution *uniforme* du type *oui/non* (*q/(1-q)*) est appliquée. Ce tirage déterminera si le prochain objet, à la position i, sera relié aux objets des positions 1 à $i-1$ [2] ou complètement indépendant de ces derniers.

2. Un objet peut être lié à lui même

Si le prochain objet devrait être indépendant, il est sélectionné par le moyen d'une distribution *Gaussienne* sur l'ensemble de toutes les instances qui ne sont pas liées avec les *i-1* objets de la séquence. Dans le cas contraire, nous avons à choisir un objet qui est relié à au moins un objet des premiers $i-1$ objets. À ce niveau, surgit la question suivante : *avec quel objet des* $i-1$ *objets le prochain objet devrait-il être relié ?* La réponse à cette question vient sous la forme d'un tirage d'une position, soit un entier de l'intervalle [*1..i*]. Dans ce tirage, une distribution qui garantit plus de chances de lier le prochain objet avec l'objet le plus proche dans la séquence devrait être utilisée. C'est ainsi que nous avons choisi d'utiliser une distribution *exponentielle* [12]. Une fois que la position k dans la séquence est déterminée, l'ensemble de tous les objets ayant un lien dans Ω avec l'objet de la position k est calculé.
Finalement, un objet est sélectionné de cet ensemble en appliquant une distribution *Gaussienne.*

Le pseudo-code du programme de génération automatique des données de test est donné par l'algorithme 23. Ce programme prend comme entrée, l'ontologie du domaine Ω, le nombre de séquences à générer et la taille minimale et maximale d'une séquence.

Algorithm 23 OSeqGen

Entrée :
Ω ; ▷ Ontologie du domaine avec respectivement $\mathscr{O}_\Omega, \mathscr{C}_\Omega$ et $\mathscr{R}_\Omega$ les objets, concepts et relations associés
$minsize \geq 1$; ▷ Taille minimale d'une séquence d'objets
$maxsize \geq minsize$; ▷ Taille maximale d'une séquence d'objets

Sortie :
s ; ▷ séquence d'objets

Initialisation :
$d \leftarrow$ *La réserve de concepts* de Ω.
$n \leftarrow PoissonDist(minsize, maxsize)$ ▷ Choix de la taille de la séquence
$l \leftarrow 1$;

Méthode :
$c \leftarrow GaussienneDist(d)$; ▷ Tirer un concept
$o \leftarrow GaussienneDist([c]_\Omega)$; ▷ Tirer un objet des instances de c
$s.\varsigma[1] \leftarrow o$;
while $(l \leq n)$ **do**
    $aLier \leftarrow UniformeDist()$; ▷ L'objet o sera-t-il lié avec un objet qui le précède ?
    if $aLier == non$ **then**
        $o \leftarrow GaussienneDist(\{o_i \in \mathscr{O}_\Omega | \forall\ j \in [1..l-1]\ \nexists\ r \in \mathscr{R}_\Omega, (s.\varsigma[j], o_i) \in [r]_\Omega\})$;
    else
        $k \leftarrow ExponentielleDist([1..l])$; ▷ Déterminer la position de l'objet avec lequel l'objet courant sera lié
        $o \leftarrow GaussienneDist(\{o_i \in \mathscr{O}_\Omega | \exists\ j \in [1..l-1], \exists\ r \in \mathscr{R}_\Omega, (s.\varsigma[j], o_i) \in [r]_\Omega\})$;
    end if
    if $o == null$ **then**
        break ;
    end if
    $s.\varsigma[l] \leftarrow o$;
    $l++$;
end while

return s ;

7.3.2.1 Jeux de données

Afin de pouvoir analyser le comportement de *xPMiner2* sous différents angles, nous avons paramétré le processus de génération de jeux de données. La procédure de génération prend en entrée trois paramètres. Il s'agit du nombre de séquences d'objets étendues à générer et des tailles minimales et maximales des séquences.

L'objectif derrière ce paramétrage des tests est de pouvoir tester l'algorithme proposé sous différentes conditions. Ainsi, les paramètres des expérimentations ont été choisis de sorte à pouvoir évaluer la mise à échelle de *xPMiner2* d'une manière similaire à celle effectuée sur l'algorithme *Apriori* (voir [10, 75]). Les configurations utilisées sont résumées dans le tableau 7.1.

Plus précisément, nous avons mené trois (différentes) séries d'expérimentations. Dans le première série de test, c'est la taille moyenne des séquences d'objets qui augmente de 10 jusqu'à 46. En revanche, le support minimal est fixé à 10% et le nombre de séquences d'objets à 100. Pour la deuxième série de tests, c'est le support qu'on fait varier de 50% à 10%, alors que nous avons utilisé un jeu de données composé de 100 séquences avec une taille moyenne de 8. Dans la troisième et dernière série de tests, le support minimal est fixé à 10%, la taille moyenne d'une séquence d'objets étendue est de 8 et le nombre de séquences d'objets étendues varie de 200 à 45000.

Résultats de la première série de tests

Le tableau 7.2 résume les résultats de la première série de tests portant sur des expérimentations de mise à l'échelle de *xPMiner2*. Dans ces expérimentations, le nombre de séquences d'objets et le support minimal sont fixes alors que la taille moyenne d'une

Groupe de tests	Nombre de séquences	taille moyenne des séquence	support minimal
Série 1	100	de 10 à 46	10%
Série 2	100	8	De 10% à 50%
Série 3	De 100 à 45000	8	10%

Tableau 7.1 – Groupes de test.

Taille moyenne	Temps d'exécution (sec)	nombre de motifs fréquents
10	0,546	3
14	1,121	5
18	2,576	16
22	52,243	225
26	69,229	249
30	2440,366	5254
34	1457,562	13053
38	36732,040	40296
42	204328,188	107302
46	490462,816	292604

Tableau 7.2 – Temps d'exécution de *xPMiner2* pour la première série de tests.

séquence augmente. En plus du temps d'exécution, le nombre de motifs fréquents découverts est rapporté dans la troisième colonne.

La courbe de la figure 7.7 reflète les données du tableau 7.2 et sera discutée plus tard en sous-section 7.3.3.

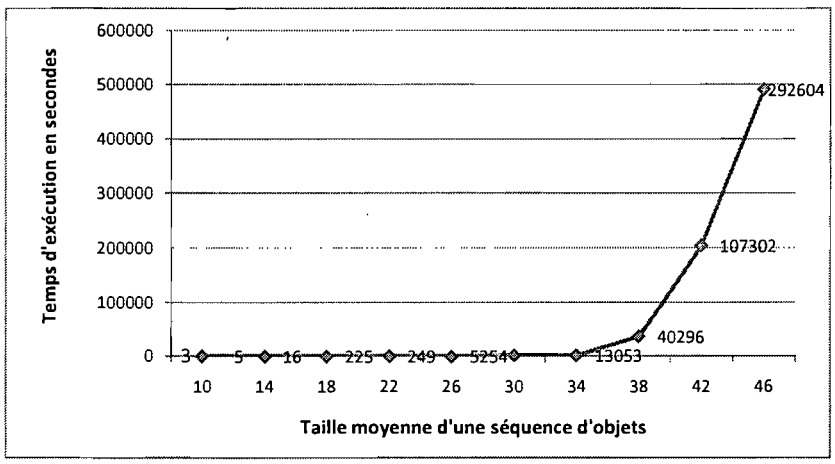


Figure 7.7 – Mise à l'échelle de l'algorithme *xPMiner2* en variant la taille moyenne des séquences d'objets.

Support	Temps d'exécution (sec)	nombre de motifs fréquents
50%	0,753	3
45%	1,940	17
40%	4,089	45
35%	8,227	98
30%	17,297	223
25%	39,888	531
20%	92,938	1337
15%	256,530	3703
10%	1223,394	18433

Tableau 7.3 – Temps d'exécution de *xPMiner2* pour la deuxième série de tests.

Résultats de la deuxième série de tests

Le tableau 7.3 résume les résultats de la deuxième série de tests. Dans cette série, le nombre de séquences d'objets et la taille de chaque séquence sont fixes alors que le support minimal varie. La troisième colonne du tableau rapporte le nombre de motifs fréquents extraits par *xPMiner2*.

La courbe de l'évolution du temps d'exécution de *xPMiner2* lorsque le support est varié est représentée dans la figure 7.8 construite à partir des données du tableau 7.3.

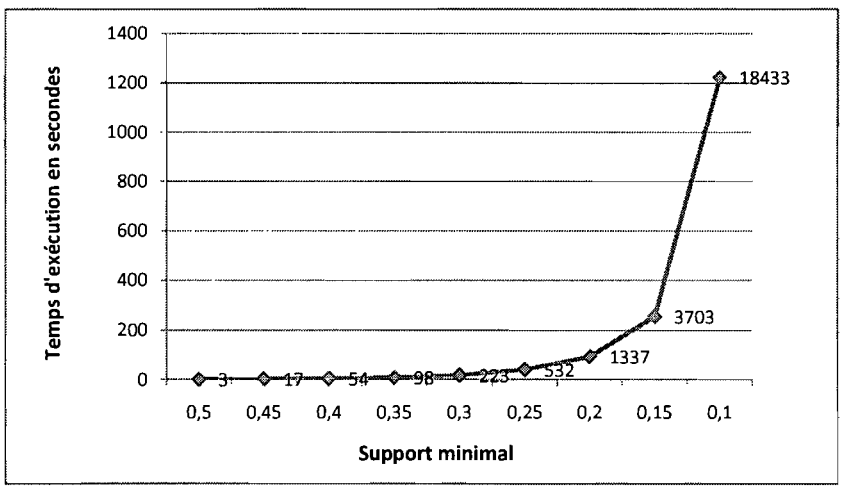


Figure 7.8 – Mise à l'échelle de l'algorithme *xPMiner2* en variant le support minimal.

Résultats de la troisième série de tests

Le tableau 7.4 résume les résultats de la dernière série de tests. Dans ces expérimentations, le nombre de séquences d'objets augmente de 100 jusqu'à 45000. En plus du temps d'exécution, le nombre de motifs fréquents découverts est rapporté dans la troisième colonne.

La figure 7.9 montre la courbe qui représente l'évolution du temps d'exécution suivant le nombre de séquences d'objets.

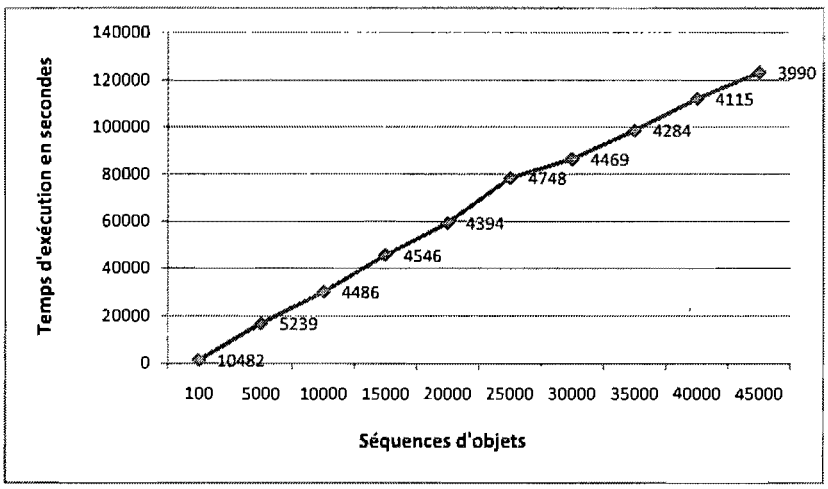


Figure 7.9 – Mise à l'échelle de l'algorithme *xPMiner2* en utilisant un nombre variable de séquences d'objets.

7.3.3 Analyse des résultats

Des deux premières courbes, nous remarquons que les performances de *xPMiner2* se détériorent rapidement avec la diminution de la valeur du support minimal et avec l'augmentation de la taille moyenne des séquences d'objets. Néanmoins, ces résultats sont de même ordre de grandeur que d'autres études qui ont porté sur la fouille des motifs séquentiels fréquents [11, 13, 60, 76, 80]. De plus, l'augmentation du temps d'exécution reflète l'augmentation exponentielle du nombre de motifs générés. À titre d'exemple,

Nombre de séquences d'objets	Temps d'exécution (sec)	nombre de motifs fréquents
100	1452,357	10482
5000	16876,514	5239
10000	30057,420	4486
15000	45453,672	4546
20000	59343,148	4394
25000	78370,648	4748
30000	86425,240	4469
35000	98712,616	4284
40000	112083,328	4115
45000	123304,488	3990

Tableau 7.4 – Temps d'exécution de *xPMiner2* pour la troisième série de tests

avec 100 séquences d'objets et une taille moyenne de 8 objets par séquence, la diminution du support minimal de 50% à 6% est accompagnée d'une augmentation considérable du nombre de motifs fréquents générés qui passe de 3 à 640540. De la troisième courbe, nous constatons que l'évolution du temps d'exécution lorsque le support minimal et la taille moyenne d'une séquence sont fixes et que le nombre de séquences varie, est quasi linéaire.

Pour mieux comprendre le comportement de l'algorithme *xPMiner2*, nous avons produit une trace de son exécution en utilisant l'outil de profilage intégré dans *NetBeans* version 6.1 [53]. Les résultats de cette analyse ont été rapportés dans la figure 7.10 et semblent indiquer que l'algorithme passe approximativement 95% du temps d'exécution à effectuer des tests d'instanciation et une grande partie de ce temps (autour de 59% du temps d'exécution de la procédure d'instanciation et 56.4% du temps total d'exécution de l'algorithme de fouille) est consacré à tester l'instanciation de concepts par des objets. Ceci est dû à la dépendance de notre implémentation à l'égard de l'API OWL de *Protégé*. En effet, nous faisons appel aux primitives de cette API pour les opérations de vérification de l'instanciation entre un concept et un objet. Or, l'appel de chacune de ces primitives nécessite un accès disque qui est coûteux en temps comparativement aux accès en mémoire principale.

Une des pistes qui pourrait aider à améliorer les performances de la procédure d'ins-

tanciation et du coup améliorer les performances de *xPMiner2*, serait de charger en mémoire l'ontologie.

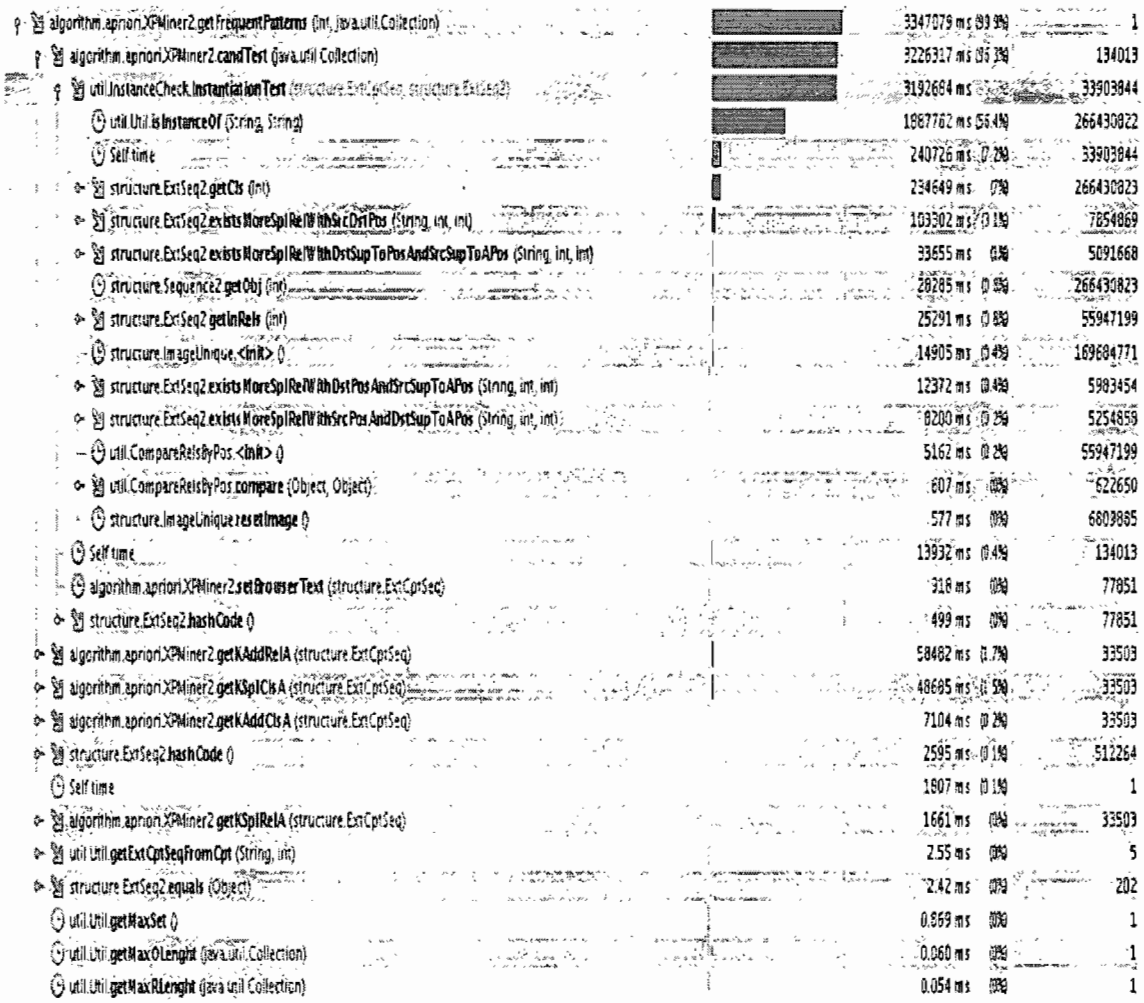


Figure 7.10 – Trace d'exécution de *xPMiner2* telle que montrée par l'outil de profilage intégré dans l'environnement de développement *NetBeans*.

7.4 Conclusion

Nous avons présenté dans ce chapitre la plateforme *OntoMiner* pour l'extraction et la manipulation des motifs séquentiels composés de concepts et de relations d'une ontologie de domaine.

Les expérimentations que nous avons menées nous ont permis de tester la mise à l'échelle de l'implémentation actuelle de l'algorithme *xPMiner2* et les résultats confirment la tendance généralement remarquée dans les algorithmes de fouille de motifs séquentiels.

Actuellement, *OntoMiner* est principalement utilisé pour simuler des sessions et extraire des motifs à partir de ces sessions. Dans une deuxième étape, le système sera intégré dans un site Web en mode production pour faire des recommandations directement aux usagers.

Chapitre 8

Conclusion générale et perspectives

Dans la présente thèse, nous avons proposé une approche de fouille de données qui intègre les connaissances du domaine lors de l'extraction de motifs séquentiels. Dans ce qui suit, nous résumons nos réalisations ainsi que nos travaux de recherche futurs.

8.1 Contribution

L'objectif de cette thèse est de construire un cadre théorique de fouille de données qui exploite les concepts et relations d'une ontologie du domaine pour découvrir des motifs séquentiels pertinents. Ces derniers peuvent être utilisés à des fins de prise de décision ou de recommandation. Pour atteindre cet objectif, nous avons défini un nouveau langage de motifs qui prend en considération les connaissances offertes par une ontologie. Ce langage est muni d'une syntaxe, d'une sémantique, et d'un ensemble de primitives pour la manipulation des différents éléments du langage.

En se basant sur les constructeurs syntaxiques et sémantiques du langage proposé, nous avons élaboré un ensemble complet de primitives permettant de passer d'un motif à un autre motif plus spécifique. Les primitives ont été par la suite utilisées pour développer une stratégie de parcours descendante (et en largeur d'abord) de l'espace des motifs suivant le degré de généralité. C'est ainsi que nous avons mis au point l'algorithme *xP-Miner2* qui se base sur plusieurs procédures pour rechercher les motifs fréquents.

Le potentiel de la nouvelle approche de génération des motifs séquentiels a été illustré via un cas de recommandation Web à travers lequel nous avons montré qu'on peut effectuer des recommandations avec une meilleure précision que les approches classiques lorsque les concepts et les relations de l'ontologie du domaine sont pris en compte.

Notre approche et nos algorithmes de fouille de motifs séquentiels ont été implémentés et testés sur le plugiciel *OntoMiner*. Les expérimentations effectuées sur des données synthétiques nous ont permis d'évaluer la mise à l'échelle de nos procédures. Les performances de notre approche sont satisfaisantes et similaires à celles des techniques existantes. Toutefois, les résultats exposés dans le chapitre 7 ont montré les limites de *xPMiner2* partiellement dues à la délégation des tâches de manipulation de l'ontologie du domaine à l'API OWL de *Protégé*. En effet, l'analyse de performances d'*OntoMiner* indiquent que l'algorithme de fouille passe approximativement 95% du temps d'exécution à effectuer des tests d'instanciation, et une grande partie de ce temps (autour de 56.4%) est consacrée à tester l'instanciation de concepts par des objets. Pour améliorer les performances, la procédure d'instanciation gagnerait à se démarquer de l'API OWL de *Protégé* et à adopter une représentation de l'ontologie qui admet des tests instantanés. Celle-ci serait chargée en mémoire au début de l'exécution, et serait optimisée pour les opérations de test d'instanciation.

Parallèlement au travail de cette thèse, nous avons exploré d'autres formes d'intégration des connaissances du domaine. C'est ainsi que nous avons proposé dans [2] un cadre pour la fouille de motifs composés de relations d'une ontologie du domaine.

8.2 Portées et limites de l'approche proposée

La fouille de motifs de concepts/relations requiert l'existence préalable d'une représentation du contenu d'un système qui inclut des hiérarchies de concepts et de relations. Cette exigence peut s'avérer contraignante pour les systèmes existants dont les données ne sont pas enrichies de connaissances.

L'autre limitation de l'approche proposée est l'absence de structure interne des objets. En effet, les objets sont considérés comme étant des unités indécomposables. Dans la pratique, des situations où des objets complexes sont manipulés peuvent se présenter. Par exemple, une image ou une vidéo peut être représentée par l'ensemble des objets qui décrivent son contenu et les relations entre eux. De même, une protèine peut être représentée par l'ensemble des acides aminés qui la composent ainsi que par les relations

existantes entre ces acides.

8.3 Perspectives

Les perspectives de recherche peuvent être classées dans les trois catégories suivantes : algorithmique, théorique et applicative.

8.3.1 Perspective algorithmique

Du point de vue algorithmique, il est possible d'améliorer les performances de l'algorithme de fouille proposé (*xPMiner2*) en adoptant une approche de fouille verticale, à l'image de celle incorporée dans l'algorithme *PrefixSpan* [41, 60, 61]. Contrairement à *Apriori*, *PrefixSpan* se passe de la génération des candidats et se base sur des divisions récursives de la base de données initiale.

8.3.2 Perspective théorique

Sur le plan de la recherche fondamentale, différentes pistes ont émergé du travail effectué dans la présente thèse, et sont regroupées en deux catégories : (1) extension du cadre de fouille proposé, et (2) développement d'un cadre générique de fouille de motifs.

8.3.2.1 Extension du cadre de fouille existant

Le cadre de motifs que nous avons proposé peut être étendu pour : (i) prendre en charge les objets complexes tels que les images et les vidéos, (ii) réduire le nombre de motifs fréquents en optant pour des représentations concises telles que les *motifs fermés* [30], les *motifs maximaux* [43], et les *méta-motifs* [81], et (iii) intégrer un module de spécification de contraintes.

Concernant ce dernier point, des contraintes/restrictions peuvent être introduites à différents niveaux. Par exemple, lors du calcul du support, nous nous sommes limités à considérer les deux cas suivants : un motif se manifeste ou ne se manifeste pas au

sein d'une même séquence d'objets étendue. Cependant, cette façon de faire ne tient pas compte du nombre de fois qu'un motif se répète au sein d'une même séquence. Or, c'est une information qui peut être intéressante pour différencier, entre autres, les motifs ayant des fréquences *inter-séquences* identiques et des fréquences *intra-séquence* différentes.

Pour en tenir compte, il est possible d'intégrer un facteur de pondération dans la valeur globale du support. Ce facteur représentera la fréquence d'apparition d'un motif au sein d'une séquence. Autrement dit, le calcul du support se basera non seulement sur le nombre de séquences d'objets couvertes par un motif, mais aussi sur le nombre d'appariements distincts au sein d'une même séquence d'objets. L'une des difficultés anticipées de cette intégration réside dans la recherche efficace de tous les appariements possibles.

Une autre contrainte qui peut être ajoutée au cadre de fouille consiste à tenir compte des distances temporelles entre deux objets et de leurs positions respectives lors de la transformation de séquences. À présent, l'extension des séquences d'objets consiste à ajouter tous les liens sémantiques, puisés de l'ontologie du domaine, qui sont co-linéaires avec l'ordre des objets dans la séquence. Afin de favoriser les liens entre les objets les plus proches dans la séquence, il est envisageable de n'ajouter que les liens entre objets qui ont été consultés/visités à l'intérieur d'une fenêtre temporelle, et à l'intérieur d'une même fenêtre temporelle considérer seulement les liens entre les objets qui sont séparés d'un maximum de n objets dans la séquence.

8.3.2.2 Vers un cadre générique de fouille de motifs

Le problème de fouille de motifs peut être défini comme la combinaison des éléments suivants : *(i)* structure des données, *(ii)* critères d'intérêt, et *(iii)* niveau d'intégration des connaissances du domaine.

Nous proposons également d'enrichir notre approche par la proposition d'un cadre générique de fouille de données qui intègre à la fois les *itemsets* (groupes d'items), les séquences et les graphes tout en exploitant une ontologie du domaine.

Ce nouveau cadre se basera sur un modèle d'intégration des connaissances qui sera

composé de deux langages : un pour la description des données et l'autre pour la description des motifs. De plus, la structure des données et des motifs sera un hypergraphe dont les noeuds et les arcs pourraient être généralisés. Pour plus de flexibilité, le cadre sera doté de différentes stratégies de parcours de l'espace de recherche et d'un module de définition de contraintes à imposer lors de la recherche de motifs. Avec un tel cadre, la définition d'une nouvelle catégorie de motifs se fera par l'instanciation du cadre générique.

8.3.3 Perspective applicative

Du point de vue applicatif, nous envisageons d'étendre *OntoMiner* de sorte à pouvoir l'intégrer dans des sites Web dont le contenu est décrit à l'aide d'ontologies du domaine. Avec cette intégration, nous allons pouvoir récupérer des données réelles et valider notre approche de recommandation qui se base sur les motifs de concepts/relations. Ainsi, il sera possible de produire des recommandations suivant les différentes approches et analyser la rétroaction des utilisateurs pour mesurer la pertinence des recommandations de chaque approche. Concrètement, un projet de plateforme de *e-tourisme* patronné par la commission du commerce et développement de l'ONU (CNUCED) est en cours de démarrage dont la composante recommandation d'itinéraires touristiques se propose d'utiliser la méthode que nous avons développée.

L'autre aspect applicatif qui mérite d'être exploré serait d'exploiter nos résultats théoriques dans des domaines autres que la recommandation Web. Le travail consistera à intégrer ces motifs dans des applications où les motifs séquentiels classiques sont utilisés et évaluer l'apport qui y résulte. Parmi ces domaines, nous citons : (i) la bioinformatique pour rechercher les motifs de séquencement de gènes dans les protéines, ou pour la découverte de motifs dans une famille de séquence de protéines (les récepteurs *cytokine*) [65], (ii) l'analyse des *logs* d'utilisation des ressources [80], et (iii) la réingénierie des ontologies en suggérant par exemple d'identifier de nouvelles relations entre les concepts qui se produisent dans les motifs sans relations explicites.

Bibliographie

[1] M. Adda, R. Missaoui, P. Valtchev et C. Djeraba. Recommendation strategy based on relation rule mining. *IJCAI Workshop on Intelligent Techniques for Web Personalization (ITWP '05).*, pages 33–40, August 2005.

[2] M. Adda, P. Valtchev et R. Missaoui. Relation rule mining. *Parallel Algorithms Appl.*, pages 439–449, 2007.

[3] M. Adda, P. Valtchev, R. Missaoui et C. Djeraba. On the discovery of semantically enhanced sequential patterns. *IEEE Proceedings of Fourth International Conference on Machine Learning and Applications (ICMLA'05).*, pages 383–390, December 2005.

[4] M. Adda, P. Valtchev, R. Missaoui et C. Djeraba. Semantically enhanced sequential patterns for content adaptation on the web. *Proceedings of MCeTech'06 : Montreal Conference on e-Technologies*, pages 177–191, May 2006.

[5] M. Adda, P. Valtchev, R. Missaoui et C. Djeraba. Toward recommendation based on ontology-powered web-usage mining. *IEEE Internet Computing Journal.*, 11 (4):45–52, July-August 2007.

[6] G. Adomavicius et A. Tuzhilin. User profiling in personalization applications through rule discovery and validation. *In Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 1999.

[7] G. Adomavicius et A. Tuzhilin. Using data mining methods to build customer profiles. *CSE 715 Seminar Personalization and Customization in E-Commerce, Bufalo University*, 2001.

[8] G. Adomavicius et A. Tuzhilin. Toward the next generation of recommender systems : a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17:734–749, June 2005.

[9] C. C. Aggarwal, J. L. Wolf, K. Wu et P. S. Yu. Horting hatches an egg : A new graph-theoretic approch to collaborative filtering. *Fifth ACM SIGKDD international conference on Knowledge discovery and data mining, California, United States*, 1999.

[10] R. Agrawal et R. Srikant. Fast algorithms for mining association rules. *In Proc. 20th International Conference on Very Large Data Bases, VLDB94*, 1994.

[11] R. Agrawal et R. Srikant. Mining sequential patterns. *In Proceedings of the Eleventh International Conference on Data Engineering, IEEE Computer Society.*, pages 3–14, 1995.

[12] Joachim H. Ahrens et Ulrich Dieter. Computer methods for sampling from gamma, beta, poisson and binomial distributions. *Computing*, 12(3):223–246, 1974.

[13] J. Ayres, J. Gehrke, T. Yiu et J. Flannick. Sequential pattern mining using a bitmap representation. *In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada*, July 2002.

[14] J. Riedl B. J. Schafer, J. A. Konstan. E-commerce recommendation applications. *Data Min. Knowl. Discov.*, 5(1-2):115–153, 2001.

[15] F. Baader, D. Calvanese, D. McGuinness, D. Nardi et P. Patel-Schneider. The description logic handbook : Theory, implementation and applications. *Cambridge University Press*, page 574, 2003.

[16] R. Baeza-Yates et B. Ribeiro-Neto. Modern information retrieval. *Addison-Wesley*, 1999.

[17] M. Baglioni, U. Ferrara, A. Romei, S. Ruggieri et F. Turini. Preprocessing and mining web log data for web personalization. *8th Italian Conf. on Artificial Intelligence*, 2829:237–249, September 2003.

[18] M. Balabanovic et Y. Shoham. Fab : Content-based, collaborative recommendation. *ACM Communication*, 40(3):66–72, 1997.

[19] S. Bandyopadhyay, U. Maulik, L.-B. Holder et D.-J. Cook. Advanced methods for knowledge discovery from complex data. *Springer Berlin Heidelberg*, pages 95–121, 2005.

[20] B. Berendt. Using and learning semantics in frequent subgraph mining. *Advances in Web Mining and Web Usage Analysis, 7th International Workshop on Knowledge Discovery on the Web, WebKDD*, pages 18–38, 2005.

[21] B. Berendt. The semantics of frequent subgraphs : Mining and navigation pattern analysis. *Workshops on Learning, Knowledge Discovery, and Adaptivity*, pages 91–102, 2006.

[22] Bettina Berendt. Using and learning semantics in frequent subgraph mining. *Workshop on Knowledge Discovery in the Web, WEBKDD*, pages 18–38, 2005.

[23] T. Berners-Lee, J. Hendler et O. Lassila. The semantic web. *Scientific American*, May 2001.

[24] C. Bettini, X.S. Wang, S. Jajodia et L. Jia-Ling. Discovering temporal relationships with multiple granularities in time sequences. *IEEE Transations on Knowledge and Data Engineering*, 10(2), 1998.

[25] Ronald J. Brachman. On the epistemological status of semantic networks. *In Nicholas V. Findler, editor, Associative Networks, Academic Press, Republished in Brachman and Levesque, 1985*, page 3Ű50, 1979.

[26] J. S. Breese, D. Hackermann et C. KADIE. analysis of predictive algorithms for collaborative filtering. *Proceeding of the fourteenth conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[27] C. Brewster, K. O'Hara, S. Fuller, Y. Wilks, E. Franconi, M. A. Musen, J. Ellman et S. B. Shum. Knowledge representation with ontologies : The present and future. *IEEE Intelligent Systems*, 19(1):72–81, 2004.

[28] I. Cadez, D. Heckerman, C. Meek, P. Smyth et S. White. Visualization of navigation patterns on a web site using model based clustering. *Technical Report MSR-TR-00-18, Microsoft Research*, 2000.

[29] J. Canny. Collaborative filtering with privacy via factor analysis. *in Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, ACM Press*, pages 238–245, 2002.

[30] Y. Chi, Y. Xia, Y. Yang et R. R. Muntz. Mining closed and maximal frequent subtrees from databases of labeled rooted trees. *UCLA Computer Science Department Technical Report CSD-TR No. 040020*, 2004.

[31] DAML Joint Commette. Daml+oil language. *http ://www.daml.org/2001/03/daml+oil-index.html*, March 2001.

[32] H. Dai et B. Mobasher. Using ontologies to discover domain-level web usage profiles. *Semantic Web Mining 2nd Workshop at ECML/PKDD, Finland*, 2002.

[33] H. Dai et B. Mobasher. Integrating semantic knowledge with web usage mining for personalization. *In Proceedings of the AAAI 2004 Workshop on Semantic Web Personalization (SWP'04)*, july 2004.

[34] M. Denny. Ontology building : A survey of editing tools, revisited. *Michigan State University*, July 2004.

[35] P. Dolog, N. Henze, W. Nejdl et M. Sintek. Towards the adaptive semantic web. *In Proc. of International Workshop on Workshop on Principles and Practice of Semantic Web Reasoning at International Conference on Logic Programming - ILP2003, Mumbai, India, Springer Verlag, LNCS 2901*, 2003.

[36] N. Drummond, A.L. Rector, R. Stevens, G. Moulton, M. Horridge, H. Wang et J. Seidenberg. Putting owl in order : Patterns for sequences in owl. *Second OWL Experiences and Directions Workshop, Athens, GA*, 2006.

[37] J. H. Gennari, S. W. Tu, T. E. Rothenfluh et M. A. Musen. Mapping domains to methods in support of reuse. *International Journal of Human-Computer Studies*, 41:399–424, 1994.

[38] G. Graef et C. Schaefer. Application of art2 networks and self-organizing maps to collaborative filtering. *7th International Workshop on Open Hypermedia Systems, Austria*, 2001.

[39] T. R. Gruber. A translation approach to portable ontology specifications. *Current issues in knowledge modeling, Academic Press Ltd. London, UK*, pages 199 – 220, 1993.

[40] D. Gunopulos, R. Khardon, H. Mannila, S. Saluja, H. Toivonen, et R. S. Sharma. Discovering all most specific sentences. *ACM Transactions on Database Systems*, 28(2):140Ű174, June 2003.

[41] J. Han et J. Pei. Mining frequent patterns by pattern-growth : methodology and implications. *ACM SIGKDD Explorations Newsletter*, 2(2):14–20, 2000.

[42] T. Horvath, B. Bringmann et L. De Raedt. Frequent hypergraph mining. *16th International Conference on Inductive Logic Programming*, pages 24–27, 2006.

[43] J. Huan, W. Wang, J. Prins et J. Yang. Spin : Mining maximal frequent subgraphs from graph databases. *in Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 581–586, 2004.

[44] A. Inokuchi, T. Washio et H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. *PKDD '00 : Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 13–23, 2000.

[45] X. Jin et B. Mobasher. Using semantic similarity to enhance item-based collaborative filtering. *In Proceedings of The 2nd IASTED International Conference on Information and Knowledge Sharing*, 2003.

[46] J. Kleinberg et M. Sandler. Convergent algorithms for collaborative filtering. *4th ACM conference on electronic commerce.*, 2003.

[47] R. Kosala et H. Blockeel. Web mining research : A survey. *SIGKDD : SIGKDD Explorations : Newsletter of the Special Interest Group (SIG) on Knowledge Discovery and Data Mining, ACM*, 2, 2000.

[48] W. Lin, S. A. Alvarez et C. Ruiz. Collaborative recommendation via adaptive association rule mining. *Worskhop on Web Mining for E-Commerce- Challenges and Opportunities, United States*, 2000.

[49] G. Manco, R. Ortale et D. Saccà. Similarity-based clustering of web transactions. *In Proceedings of the ACM symposium on Applied computing, Melbourne, Florida.*, 2003.

[50] H. Mannila, H. Toivonen et A.I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining Knowledge Discovery journal*, 1(3):259–289, July 1997.

[51] D. L. McGuinness et F. van Harmelen. Owl web ontology language overview. *http ://www.w3.org/TR/owl-features/*, 2004.

[52] P. Melville, R. J. Mooney et R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. *In Proceedings of the Eighteenth National Conference on Artificial Intelligence(AAAI-2002)*, July 2002.

[53] Sun Microsystems. Netbeans. *http ://www.netbeans.org*, 2008.

[54] Marvin Minsky. A framework for representing knowledge. *The MIT Press. In J. Haugeland, editor, Mind Design. Republished in Brachman and Levesque, 1985.*

[55] B. Mobasher, X. Jin et Y. Zhou. Semantically enhanced collaborative filtering on the web. *In Proceedings of The 2nd IASTED International Conference on Information and Knowledge Sharing*, 2004.

[56] B. Mobasher, X. Jin et Y. Zhou. Semantically enhanced collaborative filtering on the web. *In Proceedings of the European Web Mining Forum*, - 2004.

[57] P. Nokelainen, H. Tirri, M. Miettinen et T. Silander. Optimizing and profiling users online with bayesian probabilistic modeling. *In Proceedings of the NL 2002 Conference, Berlin, Germany*, 2002.

[58] P. Paulson et A. Tzanavari. Combining collaborative and content-based filtering using conceptual graphs. *Lecture Notes in Computer Science 2873 Springer*, pages 168–185, 2003.

[59] J. Pei, J. Han et R. Mao. Closet : An efficient algorithm for mining frequent closed itemsets. *ACM SIGMOD DMKD*, pages 21–30, 2000.

[60] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal et M.-C. Hsu. Prefixspan : Mining sequential patterns efficiently by prefix projected pattern growth. *In. Proc. Int. Conf. Data Engineering (ICDE'01), Heidelberg, Germany*, pages 215–226, April 2001.

[61] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal et M.-C. Hsu. Mining sequential patterns by pattern-growth : the prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16:1424–1440, Nov 2004.

[62] D. Pierrakos, G.O Paliouras, C. Papatheodorou et C.D. Spyropoulos. Web usage mining as a tool for personalization : A survey. *User Modeling and User-Adapted Interaction, Kluwer Academic Publishers, Hollande.*, 13:311–372, 2003.

[63] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen et U. Dayal. Multi-dimensional sequential pattern mining. *Proc. Int. Conf. on Information and Knowledge Management (CIKM'01), Atlanta, GA*, pages 81–88, November 2001.

[64] M. Ross Quillian. Word concepts : A theory and simulation of some basic capabilities. *Behavioral Science, Republished in Brachman and Levesque, 85*, 12: 410Ű430, 1967.

[65] G. Ramstein, P. Bunelle et Y. Jacques. Discovery of ambiguous patterns in sequences : Application to bioinformatics. pages 581–586, 2000.

[66] L. Razmerita, A. A. Angehrn et A. Maedche. Ontology-based user modeling for knowledge management systems. *The 9th International Conference on User Modeling (UM'2003)*, 2003.

[67] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom et J. Riedl. Grouplens : An open architecture for collaborative filtering of netnews. *In Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work, Chapel Hill.*, pages 175–186, 1994.

[68] C. C. Ruokangas et O. J. Mengshoel. Information filtering using bayesian networks : effective user interfaces for aviation weather data. *8th International Conference on Intelligent User Interfaces, Florida, USA*, 2003.

[69] G. Salton. Automatic text processing. *Addison-Wesley*, 1989.

[70] B. Sarwar, G. Karypis, J. Konstan et J. Riedl. Item-based collaborative filtering recommendation algorithms. *Tenth international conference on World Wide Web, Hong Kong*, 2001.

[71] B. Sarwar, G. Karypis, J. Konstan et J. Riedl. Item-based collaborative filtering recommendation algorithms. *Item-based Collaborative Filtering Recommendation Algorithms, Tenth International World Wide Web Conference, Hong Kong*, 2001.

[72] A. I. Schein, A. Popescul, L. H. Ungar et D. M. Pennock. Methods and metrics for cold-start recommendations. *In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, Finland*, pages 253–260, 2002.

[73] U. Shardanand et P. Maes. Social information filtering : Algorithms for automating "word of mouth". *In Proceedings of the SIGCHI conference on Human factors in computing systems, USA*, 1995.

[74] J.F. Sowa. Knowledge representation : Logical, philosophical, and computational foundations. *Brooks Cole Publishing Co., Pacific Grove, CA*, 2000.

[75] R. Srikant et R. Agrawal. Mining generalized association rules. *In Proceedings of the 21st International Conference on Very Large Databases (VLDB95), Zurich, Switzerland.*, September 1995.

[76] R. Srikant et R. Agrawal. Mining sequential patterns : Generalizations and performance improvements. *Proc. 5th Int. Conf. Extending Database Technology, EDBT, Avigon, France*, 1057:3–17, 1996.

[77] P. Tzvetkov, X. Yan et J. Han. Tsp : mining top-k closed sequential patterns. *Third IEEE International Conference on Data Mining, ICDM 2003, Illinois Univ., Urbana, IL, USA*, pages 347–354, Nov 2003.

[78] L. H. Ungar et D. P. Foster. Clustering methods for collaborative filtering. *Workshop on Artificial Intelligence (AAAI-98), United States*, 1998.

[79] Stanford University. Protégé. *http ://protege.stanford.edu*, 2007.

[80] W. Wang et J. Yang. *Mining Sequential Patterns from Large Data Sets (The Kluwer International Series on Advances in Database Systems).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. ISBN 0387242465.

[81] W. Wang, J. Yang et P. S. Yu. Meta-patterns : Revealing hidden periodic patterns. *In Proceedings of the 2001 IEEE International Conference on Data Mining*, 2001.

[82] Y. Xia et Y. Yang. Mining closed and maximal frequent subtrees from databases of labeled rooted trees. *IEEE Transactions on Knowledge and Data Engineering*, 17 (2):190–202, 2005.

[83] Tan Xiaoqiu, Xu Miaojun et Yao Min. An effective technique for personalization recommendation based on access sequential patterns. *In Proceedings of The IEEE Asia-Pacific Conference on Services Computing*, 17:42–46, December 2006.

[84] J. Yang, W. Wang et P. S. Yu. Discovering high order periodic patterns. *Knowledge and Information Systems, Publisher : Springer-Verlag London Ltd*, pages 243 – 268, May 2004.

[85] K. Yu, A. Schwaighofer, V. Tresp, X. Xu et H.-P. Kriegel. Probabilistic memory-based collaborative filtering. *IEEE Transactions on Knowledge and Data Engineering (TKDE), special issue on Mining and Searching the Web*, 2003.

[86] K. Yu, X. Xu1, J. Tao, M. Ester et H.-P. Kriegel. Instance selection techniques for memory-based collaborative filtering. *2nd SIAM International Conference on Data Mining, United States*, 2002.

[87] O. R. Zaiane. Building a recommender agent for e-learning systems. *International Conference on Computers in Education*, 1:55–59, 2002.

[88] M. J. Zaki. Spade : An efficient algorithm for mining frequent sequences. *Machine Learning journal*, 42(1/2):31–60, 22-24 November 2001.

[89] M. J. Zaki. Efficiently mining frequent trees in a forest. *KDD '02 : Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, Edmonton, Alberta, Canada*, pages 71–80, 2002.

[90] M.J. Zaki et C.-J. Hsiao. Charm : An efficient algorithm for closed itemset mining. *In Proceedings of the Second SIAM International Conference on Data Mining*, 2002.

[91] M.J. Zaki, N. De N. Parimi and, F. Gao, B. Phoophakdee, J. Urban, V. Chaoji, M.A Hasan et S. Salem. Towards generic pattern mining. *The Third International Conference on Formal Concept Analysis, ICFCA*, pages 1–20, 2005.

[92] Y. Zhou, X. Jin et B. Mobasher. A recommendation model based on latent principal factors in web navigation data. *In Proceedings of the 3rd International Workshop on Web Dynamics Held at the WWW 2004 Conference, New York*, May 2004.

Annexe I

Preuves de propriétés

I.1 Complétude de l'ensemble des opérations de spécialisation

La proposition 4.2.6 du chapitre 4 est composée de deux implications. La première implication consiste à dire : la génération du motif S_1 par l'application des opérations générales à un motif S_2 implique que $S_1 \sqsubseteq_{\Gamma_\Omega} S_2$. La deuxième implication quant à elle, consiste à vérifier que si $S_1 \sqsubseteq_{\Gamma_\Omega} S_2$ alors il existe au moins un chemin de calcul de cette subsomption par l'application des opérations générales. Dans la suite, nous démontrons chacune de ces implications séparément.

- **Première partie de la démonstration :** La génération du motif S_1 par l'application des opérations générales à un motif S_2 implique que $S_1 \sqsubseteq_{\Gamma_\Omega} S_2$.
  Pour démontrer que l'extension en plusieurs étapes successives d'un motif S_1 par l'application des opérations générales mène à un motif S_2 qui subsume S_1, nous procédons comme suit : en premier lieu, on fait la démonstration pour les différents cas d'extension d'un motif par l'application d'une seule opération de généralisation. Ensuite, on exploite la propriété de transitivité de la relation de subsomption (*cf.* définition 4.2.3 du chapitre 4) pour généraliser la démonstration aux cas les plus complexes (extension par l'application de plusieurs opérations successives).

  Ajout de concept Soient S_1 un motif de Γ_Ω, c un concept de $\mathscr{C}_\Omega$, et i un entier de $[1..|S_1.\varsigma|+1]$ tels que les conditions de l'application de l'opération *addCls* sont satisfaites (*cf.* définition 4.2.8 du chapitre 4), et soit S_2 le motif résultant de l'application de cette opération, c'est-à-dire, $S_2 = addCls(S_1, c, i)$

Soit la fonction $\psi : [1..|S_1.\varsigma|] \Rightarrow [1..|S_1.\varsigma|+1]$ tel que :

$$\psi(j) = \begin{cases} j \text{ si : } j < i, \\ j+1 \text{ si : } j \geq i. \end{cases}$$

De la définition de ψ, on conclut que $\forall j \in [1..|S_1.\varsigma|] : S_2.\varsigma[\psi(j)] = S_1.\varsigma[j]$, et que $\forall r(j_1, j_2) \in S_1.\theta : \exists r'(\psi(j_1), \psi(j_2)) \in S_2.\theta$ tel que $r'(\psi(j_1), \psi(j_2)) = r(j_1, j_2)$. D'où la satisfaction des conditions de la définition 4.2.6 du chapitre 4, et de ce fait $S_2 \sqsubseteq_{\Gamma_\Omega} S_1$, c'est-à-dire, $addCls(S_1, c, i) \sqsubseteq_{\Gamma\Omega} S_1$.

Ajout de relation Soient S_1 un motif de Γ_Ω, r une relation de $\mathscr{R}_\Omega$, et i, j deux entiers de $[1..|S_1.\varsigma|]$ tels que les conditions de l'application de l'opération *addRel* sont satisfaites (*cf.* définition 4.2.9 du chapitre 4), et soit S_2 le motif résultant de l'application de cette opération, c'est-à-dire, $S_2 = addRel$ (S_1, r, i, j).

On écrit $\forall k \in [1..|S_1.\varsigma|] : S_2.\varsigma[k] = S_1.\varsigma[j]$, et que $\forall r(k_1, k_2) \in S_1.\theta : \exists r'(k_1, k_2) \in S_2.\theta$ tel que $r'(k_1, k_2) = r(k_1, j2)$. Il suffit maintenant de définir la fonction injective $\psi : [1..|S_1.\varsigma|] \Rightarrow [1..|S_1.\varsigma|]$ tel que $\forall j \in [1..|S_1.\varsigma|] : \psi(j) = j$. Ainsi, on a $\forall k \in [1..|S_1.\varsigma|] : S_2.\varsigma[\psi(k)] = S_1.\varsigma[j]$, et que $\forall r(k_1, k_2) \in S_1.\theta : \exists r'(\psi(k_1), \psi(k_2)) \in S_2.\theta$ tel que $r'(\psi(k_1), \psi(k_2)) = r(k_1, k_2)$. Nous concluons que les conditions de la définition 4.2.6 du chapitre 4 sont satisfaites. Ce qui signifie que $S_2 \sqsubseteq_\Omega S_1$, c'est-à-dire, $addCls(S_1, r, i, j) \sqsubseteq_{\Gamma\Omega} S_1$.

Spécialisation de concept Soient S_1 un motif de Γ_Ω, c un concept de $\mathscr{C}_\Omega$, et i un entier de $[1..|S_1.\varsigma|]$ tels que les conditions de l'application de l'opération *splCls* sont satisfaites, c'est-à-dire qu'on a entre autres, $c \sqsubseteq_\Omega S_1.\varsigma[i]$ (*cf.* définition 4.2.7 du chapitre 4), et soit S_2 le motif résultant de l'application de l'opération en question, c'est-à-dire, $S_2 = splCls(S_1, c, i)$. Ainsi, $\forall j \in [1..|S_1.\varsigma|] : S_2.\varsigma[j] \sqsubseteq_\Omega S_1.\varsigma$, et $\forall r(j_1, j_2) \in S_1.\theta : \exists r'(j_1, j_2) \in S_2.\theta$ tel que $r'(j_1, j_2) = r(j_1, j_2)$. Maintenant, on définit la fonction $\psi : [1..|S_1.\varsigma|] \Rightarrow [1..|S_1.\varsigma|]$ tel que : $\forall j \in [1..|S_1.\varsigma|] : \psi(j) = j$. On peut écrire $\forall j \in [1..|S_1.\varsigma|] : S_2.\varsigma[\psi(j)] \sqsubseteq_\Omega S_1.\varsigma[j]$, et que $\forall r(j_1, j_2) \in S_1.\theta : \exists r'(\psi(j_1),$

$\psi(j_2)) \in S_2.\theta$ tel que $r'(\psi(j_1), \psi(j_2)) = r(j_1, j_2)$. D'où la satisfaction des conditions de la définition 4.2.6 du chapitre 4, et de ce fait : $S_2 \sqsubseteq_{\Gamma_\Omega} S_1$, c'est-à-dire, $splCls(S_1, c, i) \sqsubseteq_{\Gamma\Omega} S_1$.

Spécialisation de relation Soient S_1 un motif de Γ_Ω, r une relation de $\mathscr{R}_\Omega$, et i, j deux entier de $[1..|S_1.\varsigma|]$ tels que les conditions de l'application de l'opération *splRel* sont satisfaites, c'est-à-dire qu'on a entre autres, $\exists r'(i,j) \in S_1.\theta$ tel que $r'(i,j) \sqsubseteq_\Omega r(i,j)$ (*cf.* définition 4.2.10 du chapitre 4), et soit S_2 le motif résultant de l'application de cette opération, c'est-à-dire, $S_2 = splRel(S_1, r', r, i, j)$. Ainsi, $\forall k \in [1..|S_1.\varsigma|] : S_2.\varsigma[k] = S_1.\varsigma$, et $\forall r(k_1,k_2) \in S_1.\theta : \exists r'(k_1,k_2) \in S_2.\theta$ tel que $r'(k_1,k_2) \sqsubseteq_\Omega r(k_1,k_2)$. La fonction $\psi : [1..|S_1.\varsigma|] \Rightarrow [1..|S_1.\varsigma|]$ se définit par $\forall k \in [1..|S_1.\varsigma|] : \psi(k) = k$. On peut écrire $\forall k \in [1..|S_1.\varsigma|] : S_2.\varsigma[\psi(k)] = S_1.\varsigma[j]$, et que $\forall r(k_1,k_2) \in S_1.\theta : \exists r'(\psi(k_1), \psi(k_2)) \in S_2.\theta$ tel que $r'(\psi(k_1), \psi(k_2)) \sqsubseteq_\Omega r(k_1,k_2)$. D'où la satisfaction des conditions de la définition 4.2.6 du chapitre 4, et de ce fait : $S_2 \sqsubseteq_{\Gamma_\Omega} S_1$, c'est-à-dire, $splCls(S_1, c, i) \sqsubseteq_{\Gamma\Omega} S_1$.

Soient maintenant S et S' deux motifs tel que le S' est obtenu après l'application de n opérations générales successives à S. De ce que nous venons juste de démontrer plus haut, on peut écrire $\forall i \in [2..n] : S_i \sqsubseteq_{\Gamma_\Omega} S_{i-1}$, où $S_1 = S$ et $S_n = S'$, c'est-à-dire, $S_n = S' \sqsubseteq_{\Gamma_\Omega} S_{n-1} \sqsubseteq_{\Gamma_\Omega} S_{n-2} \ldots \sqsubseteq_{\Gamma_\Omega} S_2 \sqsubseteq_{\Gamma_\Omega} S_1 = S$. Étant donné que la relation $\sqsubseteq_{\Gamma_\Omega}$ est transitive (voir la propriété 4.2.3 du chapitre 4), on conclut que $S' \sqsubseteq_{\Gamma_\Omega} S$.

- **Deuxième partie de la démonstration :** $S_1 \sqsubseteq_{\Gamma_\Omega} S_2$ implique l'existence d'au moins un chemin de calcul de cette subsomption par l'application des opérations générales.

  Soient S_1 et S_2 deux séquences de Γ_Ω, et supposons que $S_1 \sqsubseteq_{\Gamma_\Omega} S_2$. De la définition 4.2.6 du chapitre 4, nous déduisons que :
  - $|S_2.\varsigma| \leq |S_1.\varsigma|$: le motif S_2 ne peut pas posséder plus de concepts que S_1,
  - $|S_2.\theta| \leq |S_1.\theta|$: le motif S_2 ne peut pas posséder plus de relations que S_1,
  - l'image d'un concept (resp. d'une relation) de S_1 est identique ou plus spécifique

que ce concept (resp. cette relation).

Il est à noter que pour des raisons de concision, et pour éviter toute confusion, nous allons spécifier l'ensemble de départ et d'arrivé de la fonction ψ de la définition 4.2.6 du chapitre 4 à chaque fois que le contexte change.

Nous allons procéder à la génération de S_1 à partir de S_2 en appliquant les opérations générales. Dans un premier temps, on applique les opérations de spécialisation de concept suivis des opérations de spécialisation de relation. Par la suite, on applique les opérations d'ajout de concept, et on termine par l'application des opérations d'ajout de relation. Les détails du déroulement de ces étapes et opérations sont comme suit.

À partir du motif S_2, on génère un motif S_1' par l'application d'un ensemble d'opérations de spécialisation de concept à des positions particulières de S_2 tel que :

$$\forall i \in [1..|S_1'.\varsigma|] \ : \ S_1.\varsigma[\psi(i)] = S_1'.\varsigma[i], \text{ avec } \psi : [1..|S_1'.\varsigma|] \rightarrow [1..|S_1.\varsigma|]$$

L'ordre de l'application de ces opérations est donné ci-dessous :

- $S_1' \leftarrow S_2$, on initialise S_1' par S_2,
- $\forall i \in [1..|S_1'.\varsigma|]$ si $S_1.\varsigma[\psi(i)] \sqsubset_\Omega S_1'.\varsigma[i]$ alors $S_1' \leftarrow splCls(S_1', i, S_1(\psi(i)))$, c'est-à-dire, on spécialise tous les concepts de S_2 qui ne sont pas identiques à leur image dans S_1.

Une autre série d'opérations, cette fois-ci pourtant sur la spécialisation de relation, sont appliquées au motif S_1' obtenu précédemment. L'objectif de l'application d'opérations de spécialisation de relation à S_1' est d'obtenir un motif S_2' tel que :

$$\forall r(i,j) \in S_2'.\theta \ : \ \exists r(\psi(i), \psi(j)) \in S_1.\theta, \text{ avec } \psi : [1..|S_2'.\varsigma|] \rightarrow [1..|S_1.\varsigma|]$$

L'application de ces opérations se fait dans l'ordre suivant :

- $S_2' \leftarrow S_1'$, on initialise S_2' par S_1',
- $\forall r(i,j) \in S_2'.\theta$ si $r'(\psi(i), \psi(j)) \sqsubset_\Omega r(i,j)$ alors $S_2' \leftarrow splRel(S_2', r, r', i, j)$, où

$r'(\psi(i), \psi(j)) \in S_1$, ceci revient à spécialiser toutes les relations de S_2 qui ne sont pas identiques à leur image dans S_1.

Le troisième groupe d'opérations que nous appliquons au motif résultant des deux étapes précédentes porte sur l'ajout de concept. À ce propos, le motif S_2' est transformé en S_3' par l'application d'une série d'opérations d'ajout de concept tel que :

$$\forall j \in [1..|S_1.\varsigma|] \ : \ S_3'.\varsigma[j] = S_1.\varsigma[j], \text{ avec } \psi : [1..|S_3'.\varsigma|] \rightarrow [1..|S_1.\varsigma|]$$

Ces opérations d'ajout de concept sont appliquées comme suit :
- $S_3' \leftarrow S_2'$,
- $\forall i \in ([1..|S_1.\varsigma|] - \psi([1..|S_2.\varsigma|])) : S_3' \leftarrow addCls(S_3', S_1(i), i)$, cela revient à ajouter toutes les classes qui se trouvent dans S_1 et pas dans S_2.

Le dernier groupe d'opérations que nous appliquons à S_3', porte sur l'ajout de relation pour obtenir un motif S_4' où :

$$\forall r(i,j) \in S_1.\theta \ : \ r(i,j) \in S_4'.\theta$$

Ces opérations d'ajout de relation sont appliquées comme suit :
- $S_4' \leftarrow S_3'$,
- $\forall r(i,j) \in S_1.\theta$ si $r(i,j) \notin S_4'.\theta$ alors $S_4' \leftarrow addRel(S_4', r, i, j)$, c'est-à-dire, on ajoute toutes les relations qui se trouvent dans S_1 mais pas dans S_2,

À la fin de ces quatre étapes, nous obtenons le motif S_4' qui est identique à S_1. Les différentes étapes ci-dessus sont illustrées par l'exemple donné dans la figure 4.8 du chapitre 4, où la subsomption entre les motifs S_1 et S_2 est calculée par l'exploitation des opérations générales.

Nous concluons que le motif S_1 s'obtient de S_1 par l'application des opérations de

spécialisation un nombre fini de fois. Ces opérations sont comme suit :

- ajout de $|S_1.\varsigma| - |S_2.\varsigma|$ concepts ;
- ajout de $|S_1.\theta| - |S_2.\theta|$ relations ;
- spécialisation de n concepts, où $0 \leq n \leq |S_2.\varsigma|$;
- spécialisation de p relations, où $0 \leq p \leq |S_2.\theta|$.

I.2 Ordre partiel de la relation de subsomption $\sqsubseteq_{\Gamma_\Omega}$

Pour démontrer que la relation binaire $\sqsubseteq_{\Gamma_\Omega}$ est un ordre partiel sur Γ_Ω, il est nécessaire et suffisant de démontrer que :

1. $\sqsubseteq_{\Gamma_\Omega}$ est réflexive,
2. $\sqsubseteq_{\Gamma_\Omega}$ est transitive,
3. $\sqsubseteq_{\Gamma_\Omega}$ est anti-symétrique,
4. $\exists S_1, S_2 \in (\Gamma_\Omega)^2, S_1 \not\sqsubseteq_{\Gamma_\Omega} S_2 \wedge S_2 \not\sqsubseteq_{\Gamma_\Omega} S_1$.

Les trois premiers points sont assurés par les propositions 4.2.2, 4.2.3, et 4.2.4 du chapitre 4. Il nous reste donc qu'à démontrer la validité de la quatrième expression. Pour démontrer que $\exists S_1, S_2 \in (\Gamma_\Omega)^2, S_1 \not\sqsubseteq_{\Gamma_\Omega} S_2 \wedge S_2 \not\sqsubseteq_{\Gamma_\Omega} S_1$, considérons c_1 et c_2 deux concepts distincts de l'ontologie Ω. À partir de ces deux concepts, nous construisons les deux motifs suivants : $S_1 = (\langle c_1, c_2 \rangle, \{\})$ et $S_2(\langle c_2, c_1 \rangle, \{\})$. suivant la définition 4.2.6 du chapitre 4, pour que $S_1 \sqsubseteq_{\Gamma_\Omega} S_2$, il faudrait que $c_1 \sqsubseteq_\Omega c_2$ et $c_2 \sqsubseteq_\Omega c_1$. Autrement dit, il faut avoir $c_1 = c_2$, ce qui contredit l'hypothèse de départ. D'où, $S_1 \not\sqsubseteq_{\Gamma_\Omega} S_2$. En raisonnant de manière similaire, nous arrivons à la conclusion que $S_2 \not\sqsubseteq_{\Gamma_\Omega} S_1$.

I.3 Réflexivité de la relation de subsomption

La réflexivité de $\sqsubseteq_{\Gamma_\Omega}$ est évidente du fait que cette dernière inclut la relation l'égalité de motif. En effet, $\forall S \in \Gamma_\Omega, S = S$, d'où $S \sqsubseteq_{\Gamma_\Omega} S$.

I.4 Transitivité de la relation de subsomption

Soient S_1, S_2 et S_3 trois motifs de Γ_Ω tels que $S_1 \sqsubseteq_{\Gamma_\Omega} S_2$...(I) et $S_2 \sqsubseteq_{\Gamma_\Omega} S_3$...(II). Dans la suite, nous allons utiliser les fonctions $\psi_1 : [1..|S_3.\varsigma|] \rightarrow [1..|S_2.\varsigma|]$, $\psi_2 : [1..|S_2.\varsigma|] \rightarrow [1..|S_1.\varsigma|]$ relatives, respectivement, aux processus de subsomption de S_2 par S_3 et de S_1 par S_2 (*cf.* définition 4.2.6 du chapitre 4).

De (I) et de la première condition de la définition 4.2.6 du chapitre 4, on écrit $\forall i \in [1..|S_3.\varsigma|] : S_2.\varsigma[\psi_1(i)] \sqsubseteq_\Omega S_3.\varsigma[i]$. Or, nous avons $\forall j \in [1..|S_2.\varsigma|] : S_1.\varsigma[\psi_2(j)] \sqsubseteq_\Omega S_2.\varsigma[j]$, ce qui reste valable pour tout $j \in \psi_1([1..|S_3.\varsigma|])$, c'est-à-dire, $S_1.\varsigma[\psi_2(\psi_1(i))] \sqsubseteq_\Omega S_2.\varsigma[\psi_1(i)]$. D'où, $\forall i \in [1..|S_3.\varsigma|] : S_1.\varsigma[\psi_2(\psi_1(i))] \sqsubseteq_\Omega S_3.\varsigma[i]$.
Ainsi, la première condition de la définition 4.2.6 du chapitre 4 est satisfaite.

De (II) et de la deuxième condition de la définition 4.2.6 du chapitre 4, on écrit : $\forall r(i_1,i_2) \in S_3.\theta : \exists r'(\psi_1(i_1),\psi_1(i_2)) \in S_2.\theta$ tel que $r'(\psi_1(i_1),\psi_1(i_2)) \sqsubseteq_\Omega r(i_1,i_2)$. Or, nous avons $\forall r(j_1,j_2) \in S_2.\theta : \exists r''(\psi_2(j_1),\psi_2(j_2)) \in S_1.\theta$ tel que $r''(\psi_2(j_1),\psi_2(j_2)) \sqsubseteq_\Omega r(j_1,j_2)$, ce qui est reste valable pour le sous-ensemble de relations de $S_2.\theta$ qui sont images de toutes les relations de $S_3.\theta$.
D'où, $\forall r(i_1,i_2) \in S_3.\theta : \exists r'(\psi_1(i_1),\psi_1(i_2)) \in S_2.\theta$, $\exists r''(\psi_2(\psi_1(i_1)),\psi_2(\psi_1(i_1))) \in S_1.\theta$ tel que $r'(\psi_1(i_1),\psi_1(i_2)) \sqsubseteq_\Omega r(i_1,i_2)$ et $r''(\psi_2$
$(\psi_1(i_1)),\psi_2(\psi_1(i_1))) \sqsubseteq_\Omega r'(\psi_1(i_1),\psi_1(i_2))$, ce qui implique que $r''(\psi_2(\psi_1(i_1)),\psi_2(\psi_1($
$i_1))) \sqsubseteq_\Omega r(i_1,i_2)$ pour tout $r(i_1,i_2) \in S_3.\theta$.
Ainsi, la deuxième condition de la définition 4.2.6 du chapitre 4 est satisfaite. Par conséquent, nous concluons que la relation de subsomption est transitive.

I.5 Anti-symétrie de la relation de subsomption

Soient S_1 et S_2 deux motifs de Γ_Ω tels que $S_1 \sqsubseteq_{\Gamma_\Omega} S_2$...(I) et $S_2 \sqsubseteq_{\Gamma_\Omega} S_1$...(II). Dans la suite, nous allons utiliser les fonctions $\psi_1 : [1..|S_1.\varsigma|] \rightarrow [1..|S_2.\varsigma|]$, $\psi_2 : [1..|S_2.\varsigma|] \rightarrow [1..|S_1.\varsigma|]$ relatives, respectivement, aux processus de subsomption de S_2 par S_1 et de S_1 par S_2 (*cf.* définition 4.2.6 du chapitre 4).

De (I) et de la première condition de la définition 4.2.6 du chapitre 4, on écrit $\forall i \in [1..|S_1.\varsigma|], S_2.\varsigma[\psi_1(i)] \sqsubseteq_\Omega S_1.\varsigma[i]$. De (II) et de la première condition de la définition 4.2.6 du chapitre 4, on a $\forall j \in [1..|S_2.\varsigma|], S_1.\varsigma[\psi_2(j)] \sqsubseteq_\Omega S_2.\varsigma[j]$. Ainsi, $\forall j \in [1..|S_2.\varsigma|], S_2.\varsigma[\psi_1(\psi_2(j))] \sqsubseteq_\Omega S_2.\varsigma[j]$, et $\forall i \in [1..|S_1.\varsigma|], S_1.\varsigma[\psi_2(\psi_1(i))] \sqsubseteq_\Omega S_1.\varsigma[i]$. D'où $\psi_1(\psi_2(j)) = j$ et $\psi_2(\psi_1(i)) = i$, ce qui est équivalent à dire que $\psi_1 o \psi_2 = Id_{[1..|S_1.\varsigma|]}$, et $\psi_2 o \psi_1 = Id_{[1..|S_2.\varsigma|]}$ où Id_X est la fonction identité par rapport à l'ensemble X. Autrement dit, ψ_2 est la fonction réciproque de ψ_1, et par conséquent ψ_1 et ψ_2 sont des bijections. Ainsi, $|S_1| = |S_2|$, et de ce fait, $\forall j \in [1..|S_2.\varsigma|], \psi_2(j) = j$, et $\forall i \in [1..|S_1.\varsigma|], \psi_1(i) = i$. Maintenant, les expressions $\forall i \in [1..|S_1.\varsigma|], S_2.\varsigma[\psi_1(i)] \sqsubseteq_\Omega S_1.\varsigma[i]$, et $\forall j \in [1..|S_2.\varsigma|]$, $S_1.\varsigma[\psi_2(j)] \sqsubseteq_\Omega S_2.\varsigma[j]$ sont réécrites par $\forall i \in [1..|S_1.\varsigma|], S_2.\varsigma[i] \sqsubseteq_\Omega S_1.\varsigma[i]$, et $\forall j \in [1..|S_2.\varsigma|], S_1.\varsigma[j] \sqsubseteq_\Omega S_2.\varsigma[j]$. Ce qui est équivalent à dire que $\forall i \in [1..|S_1.\varsigma|], S_2.\varsigma[i] = S_1.\varsigma[i]$.

De (I) et de la deuxième condition de la définition 4.2.6 du chapitre 4, on écrit $\forall r(i_1, i_2) \in S_1.\theta, \exists r'(\psi_1(i_1), \psi_1(i_2)) \in S_2.\theta$ tel que $r'(\psi_1(i_1), \psi_1(i_2)) \sqsubseteq_\Omega r(i_1, i_2)$. Comme nous venons de prouver que $\forall i \in [1..|S_1.\varsigma|] \psi_1(i) = i$, alors $\forall r(i_1, i_2) \in S_1.\theta, \exists r'(i_1, i_2) \in S_2.\theta$ tel que $r'(i_1, i_2) \sqsubseteq_\Omega r(i_1, i_2)$...(a).

De (II) et de la deuxième condition de la définition 4.2.6 du chapitre 4, on écrit $\forall r''(j_1, j_2) \in S_2.\theta, \exists r'''(\psi_2(j_1), \psi_2(j_2)) \in S_1.\theta$ tel que $r'''(\psi_2(j_1), \psi_2(j_2)) \sqsubseteq_\Omega r''(j_1, j_2)$. Ceci peut être réécrit comme suit : $\forall r''(j_1, j_2) \in S_2.\theta, \exists r'''(j_1, j_2) \in S_1.\theta$ tel que $r'''(j_1, j_2) \sqsubseteq_\Omega r''(j_1, j_2)$...(b). De (a) et (b), on écrit $\forall r(i_1, i_2) \in S_1.\theta, \exists r''(i_1, i_2) \in S_2.\theta, \exists r'''(i_1, i_2) \in S_1.\theta$ tel que $r'''(i_1, i_2) \sqsubseteq_\Omega r''(i_1, i_2) \sqsubseteq_\Omega r(i_1, i_2)$. Supposons maintenant qu'il existe une relation $r_1^1(i_1, i_2) \in S_1.\theta$ tel que $r_1^1(i_1, i_2) \notin S_2.\theta$. Ainsi, l'image correspondante dans $S_2.\theta$, notée $r_1^2(i_1, i_2)$ sera différente de $r_1^1(i_1, i_2)$. De même, l'image dans $S_1.\theta$ qui sera

associée à $r_1^2(i_1,i_2)$, notée $r_2^1(i_1,i_2)$ sera aussi différente de $r_1^1(i_1,i_2)$, sinon, on aura $r_1^2(i_1,i_2) = r_1^1(i_1,i_2)$ ce qui contredit l'hypothèse de départ. Donc, nous avons $r_1^1(i_1,i_2) \sqsubseteq_\Omega r_1^2(i_1,i_2) \sqsubseteq_\Omega r_2^1(i_1,i_2)$. Soit $r_2^2(i_1,i_2)$ l'image de $r_2^1(i_1,i_2)$ dans $S_2.\theta$. Maintenant, on sais que l'image de $r_2^2(i_1,i_2)$ dans $S_1.\theta$ est différente de $r_2^1(i_1,i_2)$. Car sinon, on aura $r_2^1(i_1,i_2)$ comme étant l'image des deux relations $r_2^1(i_1,i_2)$ et $r_2^2(i_1,i_2)$, ce qui n'est pas autorisé par la définition de la subsomption. Cette nouvelle image sera notée $r_1^3(i_1,i_2)$. Le même raisonnement est récursivement appliqué à $r_1^3(i_1,i_2)$ et aux images obtenues dans chaque étape. Lorsque tous les éléments de $S_1.\theta$ et de $S_2.\theta$ ont été épuisés, tous les éléments de $S_1.\theta$ leur est associé une image dans $S_2.\theta$. De même pour chaque élément de $S_2.\theta$ est associé une image dans $S_1.\theta$, sauf pour la relation $r_2^{|S_1.\theta|}(i_1,i_2)$, car sinon, un élément de $S_1.\theta$ sera l'image de deux relations de $S_2.\theta$, ce qui n'est pas autorisé par la définition de la subsomption. Le seul élément de $S_1.\theta$ qui n'est l'image d'aucun élément de $S_2.\theta$, est la première relation $r_1^1(i_1,i_2)$. Or, nous avons par transitivité $r_1^1(i_1,i_2) \sqsubseteq_\Omega r_2^{|S_1.\theta|}(i_1,i_2)$ et donc on ne peut avoir $r_1^1(i_1,i_2)$ comme étant l'image de $r_2^{|S_1.\theta|}(i_1,i_2)$ que si $r_2^{|S_1.\theta|}(i_1,i_2) \sqsubseteq_\Omega r_1^1(i_1,i_2)$. Ce qui nous amène à dire que $r_2^{|S_1.\theta|}(i_1,i_2) = r_1^1(i_1,i_2)$. Ceci est en contradiction avec l'hypothèse de départ qui stipule qu'il n'existe pas de relation dans $S_2.\theta$ qui est identique à $r_1^1(i_1,i_2)$. Nous concluons que toute relation de $S_1.\theta$ existe dans $S_2.\theta$, et comme la cardinalité de $S_2.\theta$ est égale à celle de $S_1.\theta$, alors $S_2.\theta = S_1.\theta$.

Ainsi, nous concluons que $S_1 = S_2$.

I.6 Subsomption et généralisation de motifs

Démontrer la proposition 4.2.5 du chapitre 4, revient à démontrer que l'expression $(S_1 \sqsubseteq_{\Gamma_\Omega} S_2 \Rightarrow S_1 \leq S_2)$ est vraie.

Il est à signaler qu'on ce qui concerne les instances des concepts et des relations d'une ontologie, nous considérons le cas général. En d'autres termes, on ne réduit pas la définition par extension d'un concept (resp. d'une relation) au sous-ensemble d'objets (resp. de couples d'objets) disponibles, mais on tient compte de l'ensemble de toutes les instances possibles même si concrètement on n'en dispose que d'un sous-ensemble de ces instances.

Soient S_1 et S_2 deux motifs de Γ_Ω, afin de démontrer que si $S_1 \sqsubseteq_{\Gamma_\Omega} S_2$, alors $S_1 \leq S_2$, nous procédons par la composition des fonctions d'instanciation et de subsomption. Supposons que $S_1 \sqsubseteq_{\Gamma_\Omega} S_2$ et $s = (\langle o_i \rangle_{i=1..|s.\varsigma|}, \{r(l,m) | 1 \leq l \leq m \leq n\})$ est une séquence d'objets étendue tel que $s \in [S_1]_{\Gamma_\Omega}$, et montrons que $s \in [S_2]_{\Gamma_\Omega}$. Pour cela, nous devons montrer que les deux conditions de la définition 4.2.3 du chapitre 4 sont satisfaites. Dans la suite, nous allons utiliser les fonctions $\phi_1 : [1..|S_1.\varsigma|] \rightarrow [1..|s.\varsigma|]$, $\phi_2 : [1..|S_2.\varsigma|] \rightarrow [1..|s.\varsigma|]$ relatives, respectivement, aux processus d'instanciation de S_1 par s et de S_2 par s (*cf.* définition 4.2.3 du chapitre 4).

Commençons par la première condition de la définition 4.2.3 du chapitre 4. Soit l'ensemble $I \subseteq [1..|S_1.\varsigma|]$ définit par $I = \{\psi(i) \in [1..|S_1.\varsigma|] | i \in [1..|S_2.\varsigma|]\}$ et qui peut être réécrit par $I = \psi([1..|S_2.\varsigma|])$, avec $\psi()$ la fonction de mise en correspondance présentée dans la définition 4.2.6 du chapitre 4.
Nous avons aussi $\forall i' \in [1..|S_1.\varsigma|] \;\; o_{\phi(i')} \in [S_1.\varsigma[i']]_\Omega$ (*cf.* définition 4.2.3 du chapitre 4). Ceci reste vrai pour tout $i' \in I$. Or, $\forall i' \in I$ nous avons $S_1.\varsigma[i'] \sqsubseteq_\Omega S_1.\varsigma[\psi^{-1}(i')]$. D'où, $\forall i \in [1..|S_2.\varsigma|]$, nous avons $o_{\phi(\psi(i))} \in [S_2.\varsigma[i]]_\Omega$. Ce qui montre que pour s et S_2, la première condition de la définition 4.2.3 du chapitre 4 est satisfaite.

Pour démontrer la seconde partie de la définition 4.2.3 du chapitre 4, considérons $I = \psi([1..|S_2.\varsigma|])$ (emprunté de la première partie de la démonstration). D'un côté, nous avons $\forall r(i_1,i_2) \in S_2.\theta, \; \exists \; (j_1,j_2) \in ([1..|S_1.\varsigma|])^2 \; : \; \psi(i_1) = j_1, \; \psi(i_2) = j_2$ et

$\exists\ r'(j_1, j_2) \in S_1.\theta$ tel que $r' \sqsubseteq_\Omega r$ (deuxième condition de la définition 4.2.6 du chapitre 4). D'un autre côté, nous avons $\forall r''(j_1, j_2) \in S_1.\theta, \exists\ (k_1, k_2) \in ([1..|s.\varsigma|])^2$ tels que $\phi(j_1) = k_1, \phi(j_2) = k_2$, et $\exists\ r'''(k_1, k_2) \in s.\theta$, tel que $r''' \sqsubseteq_\Omega r''$ et $(o_{k_1}, o_{k_2}) \in [r''(j_1, j_2)]_\Omega$ (deuxième condition de la définition 4.2.3 du chapitre 4). D'où $\forall r(i_1, i_2) \in S_2.\theta$, $\exists\ (k_1, k_2) \in ([1..|s.\varsigma|])^2\ :\ \phi(\psi(i_1)) = k_1,\ \phi(\psi(i_2)) = k_2$ et $\exists\ r'''(k_1, k_2) \in s.\theta$ tel que $r''' \sqsubseteq_\Omega r$ et $(o_{k_1}, o_{k_2}) \in [r(i_1, i_2)]_\Omega$.

Nous concluons que $s \in [S_2]_\Omega$, et par conséquent, $[S_1]_\Omega \subseteq [S_2]_\Omega$ implique que $S_1 \leq S_2$.

I.7 La monotonie du rang

Pour montrer que τ est une fonction monotone décroissante par rapport à la relation de généralité, on doit démontrer que l'application de chaque opération de spécialisation à S (ajout de concept, ajout de relation, spécialisation de concept, et spécialisation de relation) donnera lieu à une séquence qui a un rang strictement supérieur.
Dans la suite nous considérons τ_1 le rang associé à la S_1.

a) Spécialiser un concept : Considérons le concept c, et j un entier tel que S_1 satisfait les conditions de la définition 13 du chapitre 5.

La séquence $S_2 = splCls(S_1, c, j)$ est obtenue par l'application d'une telle opération de spécialisation, et soit $\tau_2 = \tau(S_2)$ son rang. Nous avons, $\tau_2 = \sum_{c' \in S_2.\varsigma} P_c(c') + \sum_{r \in S_2.\theta} P_r(r)$. De ces expressions, nous écrivons :

1. $P_c(c) = P_c(S_1.\varsigma[j]) + a$, où a est un entier tel que $a \geq 1$;
2. Étant donné que $S_2.\varsigma[i] = S_2.\varsigma[j]$ pour tout $i \neq j$, alors $\sum_{c' \in (S_2.\varsigma - \{c\})} P_c(c') = \sum_{c' \in (S_2.\varsigma - \{S_1.\varsigma[j]\})} P_c(c')$;
3. $\sum_{r \in S_1.\theta} P_r(r) = \sum_{r \in S_2.\theta} P_r(r)$ parce que $S_1.\theta = S_2.\theta$.

Ainsi, nous avons $P_c(c) + \sum_{c' \in (S_2.\varsigma - \{c\})} P_c(c') = P_c(S_1.\varsigma[j]) + a + \sum_{c' \in (S_2.\varsigma - \{S_1.\varsigma[j]\})} P_c(c')$. Ce qui est équivalent à : $\sum_{c' \in S_2.\varsigma} P_c(c') = a + \sum_{c' \in S_1.\varsigma} P_c(c')$. Par conséquent, $\sum_{c' \in S_2.\varsigma} P_c(c') + \sum_{r \in S_2.\theta} P_r(r) = \sum_{c' \in S_1.\varsigma} P_c(c') + \sum_{r \in S_1.\varsigma} P_r(r) + a$. Ce qui nous conduit à conclure que $\tau_2 = \tau_1 + a$, impliquant que $\tau_2 > \tau_1$.

b) Ajout de concept : Soit le concept c, et j un entier tel que S_1 satisfait les conditions de la définition 11 du chapitre 5.

La séquence $S_2 = addCls(S_1, c, j)$ est obtenue par l'application de l'opération en question (ajout de classe), et nous considérons son rang $\tau_2 = \tau(S_2)$. Nous avons $\tau_2 = \sum_{c' \in S_2.\varsigma} P_c(c') + \sum_{r \in S_2.\theta} P_r(r)$. De ce qui précède et de la définition 11 du chapitre 5, on écrit :

1. $\sum_{c' \in S_2.\varsigma} P_c(c') = \sum_{c' \in S_1.\varsigma} P_c(c') + b$, où b est un entier tel que $b = P_c(c)$ et $b \geq 1$;

2. $\sum_{r \in S_1.\theta} P_r(r) = \sum_{r \in S_2.\theta} P_r(r)$.

Des déclarations 1) et 2) nous pouvons écrire $\sum_{c' \in S_2.\varsigma} P_c(c') + \sum_{r \in S_2.\theta} P_r(r) = \sum_{c' \in S_1.\varsigma} P_c(c') + \sum_{r \in S_1.\theta} P_r(r) + b$. D'où, $\tau_2 = \tau_1 + b$, et par conséquence $\tau_2 > \tau_1$.

c) Ajout d'une relation : Considérons la relation r, et i, j deux entiers tel que S_1 satisfait les conditions de la définition 12 du chapitre 5.

La séquence $S_2 = addRel(S_1, r, i, j)$ est obtenue par l'application de l'opération d'insertion de relation, et nous considérons son rang $\tau_2 = \tau(S_2)$. Nous avons $\tau_2 = \sum_{c' \in S_2.\varsigma} P_c(c') + \sum_{r \in S_2.\theta} P_r(r)$. De ces expressions et de la définition 12 du chapitre 5, nous écrivons :

1. $\sum_{c' \in S_1.\varsigma} P_c(c') = \sum_{c' \in S_2.\varsigma} P_c(c')$;
2. $\sum_{r \in S_2.\theta} P_r(r) = \sum_{r \in S_1.\theta} P_r(r) + d$, où d est un entier tel que $d = P_r(r)$ et $d \geq 1$;

De 1) et 2) nous écrivons $\sum_{c' \in S_2.\varsigma} P_c(c') + \sum_{r \in S_2.\theta} P_r(r) = \sum_{c' \in S_1.\varsigma} P_c(c') + \sum_{r \in S_1.\theta} P_r(r) + d$, qui est équivalent à : $\tau_2 = \tau_1 + d$. Ainsi, nous avons $\tau_2 > \tau_1$.

d) Spécialiser une relation : Considérons la relation r, et j un entier tel que S_1 satisfait les conditions de la définition 14 du chapitre 5.

La séquence $S_2 = splRel(S_1, r_1, i, j, r_2)$ est obtenue par l'application de l'opération de spécialisation de relation tel que spécifié, et nous considérons son rang $\tau_2 = \tau(S_2)$. Nous avons $\tau_2 = \sum_{c' \in S_2.\varsigma} P_c(c') + \sum_{r \in S_2.\theta} P_r(r)$. De ce qui précède et de la définition 14 du chapitre 5, nous avons :

1. $\sum_{c' \in S_1.\varsigma} P_c(c') = \sum_{c' \in S_2.\varsigma} P_c(c')$;
2. $P_r(r_2) = P_r(r_1) + e$, où e est un entier tel que $e \geq 1$;
3. $\sum_{r \in (S_2.\theta - r_2(i,j))} P_r(r) = \sum_{r \in (S_1.\theta - r_1(i,j))} P_r(r)$ parce que $S_1.\theta = S_2.\theta$.

De 2) et 3) nous avons $\sum_{r \in S_2.\theta} P_r(r) = \sum_{r \in S_1.\theta} P_r(r) + e$, et ainsi $\sum_{c' \in S_2.\varsigma} P_c(c') + \sum_{r \in S_2.\theta} P_r(r) = \sum_{c' \in S_1.\varsigma} P_c(c') + \sum_{r \in S_1.\theta} P_r(r) + e$. Par conséquent, nous pouvons écrire $\tau_2 = \tau_1 + e$, ce qui implique que $\tau_2 > \tau_1$.

I.8 Complétude de l'ensemble des opérations canoniques

Soit $\mathscr{F}_{\Omega}^{k}$ l'ensemble de tous les motifs de rang k. La proposition 5.1.2 du chapitre 5 est reformulée comme suit : $P(k) : \forall S \in \mathscr{F}_{\Omega}^{k}, \exists\, S' \in \mathscr{F}_{\Omega}^{k-1} : S = ops(S')$ où $k \geq 1$ et $ops()$ correspond à l'une des quatre opérations canoniques ($addClsC(), addRelC(), splClsC(), splRelC()$). Nous utilisons un raisonnement par récurrence sur la propriété $P(k)$ pour démontrer que la proposition 5.1.2 du chapitre 5 est vraie, ce qui revient à démontrer que $P(k)$ est vraie pour tout entier naturel $k > 0$.

Initialisation : $k = 1$

La proposition $P(1)$ est équivalente à dire que tout motif de niveau 1 peut être généré à partir d'au moins un motif de niveau 0 par l'application d'une opération canonique. Le niveau 0 contient un seul motif et c'est le motif vide $\emptyset_{\Gamma_{\Omega}}$. Pour que $P(1)$ soit vérifiée, tous les motifs de niveau 1 (motifs singletons) devraient être générés par les opérations canoniques à partir de $\emptyset_{\Gamma_{\Omega}}$. Or, pour tout concept c de $\mathscr{C}_{\Omega}$ où $P_c(c) = 1$ on a $S = (\langle c \rangle, \{\})$ $\in \mathscr{F}_{\Omega}^{k}$. Ainsi, et suivant la définition de l'opération d'ajout canonique de concept (*cf.* définition 5.1.11 du chapitre 5), $S = addClsC(\emptyset_{\Gamma_{\Omega}}, c)$. D'où, $P(k)$ est vérifiée pour $k = 1$...*(I)*

Induction : pour tout $k > 1$

Supposons que la propriété $P(k)$ est vraie pour tout entier naturel $k \geq 1$, et montrons qu'elle reste vraie pour $k+1$. $P(k)$ vraie pour $k \geq 1$ signifie que tout motif de niveau k peut être généré par au moins un motif de niveau $k-1$.

Pour démontrer que $P(k+1)$ est vraie, nous allons procéder par l'absurde. Autrement dit, supposons qu'il existe un motif S de rang $k+1$ ($\tau(S') = k+1$) qui ne peut pas être généré à partir des motifs de rang k par l'application d'une opération canonique. Le motif S peut être écrit de la manière suivante : $S = (\langle c_1, c_2, ..., c_{m-1}, c_m \rangle, \theta)$. Suivant la profondeur de la dernière classe et des relations de S qui aboutissent à la dernière

position, les différents cas à distinguer sont :

1. $\nexists\, r(i,m) \in S.\theta$, c'est-à-dire qu'aucune relation de S n'a comme cible la classe c_m. Suivant la hauteur de c_m, deux cas sont à distinguer :
   (a) $P_c(c_m) = 1$: alors $\exists\, S' \in \in \mathscr{F}^k_\Omega$ tel que $S' = (\langle c_1, c_2, ..., c_{m-1}\rangle, S.\theta) \in \mathscr{F}^n_\Omega$. D'où $S = addClsC(S', c_m)$;
   (b) $P_c(c_m) > 1$: alors $\exists\, c'_m \in \mathscr{C}_\Omega$ tel que $c_m \sqsubseteq^r_\Omega c'_m$. Donc $\exists\, S' \in \in \mathscr{F}^k_\Omega$ tel que $S' = (\langle c_1, c_2, ..., c'_m\rangle, S.\theta) \in \mathscr{F}^n_\Omega$. D'où $S = splClsC(S', c'_m, c_m)$;
2. $\exists\, r(i,m) \in S.\theta$ où $i < m$, c'est-à-dire qu'il existe au moins une relation partant d'une classe de S et aboutissant à c_m. Suivant la profondeur de la relation r dans l'ontologie Ω, deux cas sont à distinguer :
   (a) $P_r(r) = 1$: alors $\exists\, S' \in \Gamma_\Omega$ tel que $S' = (\langle c_1, c_2, ..., c_m\rangle, S.\theta - \{r(i,m)\}) \in \mathscr{F}^k_\Omega$. Ainsi, $S = addRelC(S', r, i)$;
   (b) $P_r(r) > 1$: alors $\exists\, r' \in \mathscr{R}_\Omega$, $r \sqsubseteq^r_\Omega r'$ tel que :
   - $c_i \ll dom(r')$ car $c_i \ll dom(r)$ et $r \sqsubseteq^r_\Omega r'$;
   - $c_m \ll ran(r', c_i)$ car $c_m \ll ran(r, c_i)$ et $r \sqsubseteq^r_\Omega r'$;

   alors $\exists\, S' \in \Gamma_\Omega$ tel que $S' = (\langle c_1, c_2, ..., c_m\rangle, (S.\theta - \{r(i,m)\}) \cup \{r'(i,m)\}) \in \mathscr{F}^k_\Omega$. Ainsi, les conditions de l'application de la définition 5.1.13 du chapitre 5 sont satisfaites. D'où, $S = splRelC(S', r, i, r')$.

Étant donné que dans tous les cas une contradiction a été retrouvée avec l'hypothèse de départ, nous concluons que la propriété $P(k+1)$ est vraie...*(II)*

À partir des conclusions tirées des étapes d'initialisation *(I)* et d'induction *(II)*, nous affirmons que la propriété $P(k)$ est vraie pour tout $k \geq 1$.

I.9 Relation d'ordre total

Soient A et B deux éléments de $\mathscr{C}_\Omega \cup \rho$. Nous distinguons trois différents cas :

1. $A,B \in \mathscr{C}_\Omega^2$: en se basant sur la définition 5.2.1 du chapitre 5, nous avons ce qui suit :
   - si $\mathscr{L}(A) \leq_{lex} \mathscr{L}(B)$ alors $A \leq B$,
   - si $\mathscr{L}(B) \leq_{lex} \mathscr{L}(A)$ alors $B \leq A$,
2. $A \in \mathscr{C}_\Omega \wedge B \in \rho : A \leq B$ (de la définition 5.2.1 du chapitre 5),
3. $A,B \in \rho^2$ **avec** $A = r_1(l_1,m_1), B = r_2(l_2,m_2)$: suivant la définition 5.2.1 du chapitre 5, on peut écrire :
   - si $\mathscr{L}(r_1) <_{lex} \mathscr{L}(r_2)$ alors $A \leq B$,
   - si $\mathscr{L}(r_2) <_{lex} \mathscr{L}(r_1)$ alors $B \leq A$,
   - si $\mathscr{L}(r_1) =_{lex} \mathscr{L}(r_2) \wedge (l_1 < l_2 \vee (l_1 = l_2 \wedge m_1 \leq m_2))$ alors $A \leq B$,
   - si $\mathscr{L}(r_1) =_{lex} \mathscr{L}(r_2) \wedge (l_1 > l_2 \vee (l_1 = l_2 \wedge m_1 \geq m_2))$ alors $B \leq A$,

Nous concluons que dans tous les cas A et B sont comparables suivant la relation d'ordre $\leq$.

I.10 Complétude de l'ensemble des opérations de suffixe

Soit $\mathscr{F}_{\Omega}^{k}$ l'ensemble de tous les motifs de rang k. La proposition 5.2.2 du chapitre 5 est reformulée comme suit : $P(k) : \forall S \in \mathscr{F}_{\Omega}^{k}, \exists\, S' \in \mathscr{F}_{\Omega}^{k-1} : S = ops(S')$ où $k \geq 1$ et $ops()$ correspond à l'une des quatre opérations de suffixe ($addClsA(), addRelA(), splClsA(), splRelA()$). Nous utilisons un raisonnement par récurrence sur la propriété $P(k)$ pour démontrer que la proposition 5.2.2 du chapitre 5 est vraie, ce qui revient à démontrer que $P(k)$ est vraie pour tout entier naturel $k > 0$.

Initialisation : $k = 1$

La proposition $P(1)$ est équivalente à dire que tout motif de niveau 1 peut être généré par l'application d'une opération de suffixe sur un motif de niveau 0. Le niveau 0 contient un seul motif et c'est le motif vide $\emptyset_{\Gamma_{\Omega}}$. Pour que $P(1)$ soit vérifiée, il faudrait que tous les motifs de niveau 1 (motifs singletons) puissent être générés par l'application des opérations de suffixe sur $\emptyset_{\Gamma_{\Omega}}$. Or, pour tout concept c de $\mathscr{C}_{\Omega}$, où $P_c(c) = 1$, nous avons $S = (\langle c \rangle, \{\}) \in \mathscr{F}_{\Omega}^{k}$. Ainsi, et suivant la définition de l'opération de suffixe pour l'ajout de concept, $S = addClsA(\emptyset_{\Gamma_{\Omega}}, c)$ (*cf.* définition 5.2.6 du chapitre 5). D'où, $P(k)$ est vérifiée pour $k = 1$...*(I)*

Induction : pour tout $k > 1$

Supposons que la propriété $P(k)$ est vraie pour un entier naturel $k \geq 1$, et montrons qu'elle reste vraie pour $k+1$. $P(k)$ vraie pour $k \geq 1$ signifie que tout motif de niveau k peut être généré par au moins un motif de niveau $k-1$.

Pour démontrer que $P(k+1)$ est vraie, nous allons procéder par l'absurde. Autrement dit, supposons qu'il existe un motif $S = (\langle c_1, c_2, ..., c_{m-1}, c_m \rangle, \theta)$ de rang $k+1$ ($\tau(S') = k+1$) qui ne peut pas être généré à partir d'un motif de rang k par l'application d'une opération de suffixe. Suivant la profondeur de la dernière classe et des relations de S qui aboutissent à la dernière position, les différents cas à distinguer sont :

1. $\nexists\, r(i,m) \in S.\theta$, c'est-à-dire qu'aucun triplet de S n'a comme cible la classe c_m. Suivant la hauteur de c_m, deux cas sont à distinguer :

   (a) $P_c(c_m) = 1$: étant donné que $P(k)$ est vérifiée, alors $\exists\, S' \in \mathscr{F}_\Omega^k$ tel que $S' = (\langle c_1, c_2, ..., c_{m-1}\rangle, S.\theta)$. Ainsi, les conditions de l'application de l'opération de suffixe pour d'ajout de concept sont satisfaites, d'où $S = addClsA(S', c_m)$ (*cf.* définition 5.2.6 du chapitre 5) ;

   (b) $P_c(c_m) > 1$: soit $c' \in \mathscr{C}_\Omega$ tel que c' est l'élément le plus grand de l'ensemble $L = \{c'' \in \mathscr{C}_\Omega | c_m \sqsubseteq_\Omega^r c''\}$. C'est-à-dire, $\forall c \in L - \{c'\}$ on a $c <_c c'$. Soit le motif $S' \in \mathscr{F}_\Omega^k$ tel que $S' = (\langle c_1, c_2, ..., c_{m-1}, c'\rangle, S.\theta)$. Ainsi, $\Psi_c^n(c_m, S') = L - \{c'\}$ (*cf.* définition 5.2.3 du chapitre 5). Comme le concept c' est plus grand que tous les éléments de $\Psi_c^n(c_m, S')$, alors les conditions de l'application de l'opération de suffixe pour la spécialisation de concept sont satisfaites. D'où $S = splClsA(S', c')$ (*cf.* définition 5.2.7 du chapitre 5).

2. $\exists\, r(i,m) \in S.\theta$, c'est-à-dire qu'il existe au moins un triplet dans S qui fait intervenir la dernière position. Les deux cas ci-dessous se présentent.

   (a) **Il existe au moins un triplet qui fait intervenir la position m et dont la relation est une racine :** Soient $r'(i', m)$ le plus grand triplet de l'ensemble $L = \{r''(i'', m) \in S.\theta | P_r(r'') = 1\}$, et S' un motif de $\mathscr{F}_\Omega^k$ tel que $S' = (S.\varsigma, \theta - \{r'(i', m)\})$. De la définition 5.2.4 du chapitre 5, nous écrivons $\Psi_r^1(r'(i', m), S') = L - \{r'(i', m)\}$. Comme $r'(i', m)$ est le plus grand élément de L, alors $r'(i', m)$ reste le plus grand élément de $\Psi_r^1(r'(i', m), S') = L - \{r'(i', m)\}$. Ainsi, les conditions de l'application de l'opération de suffixe pour l'ajout de relation sont satisfaites, d'où $S = addRelA(S', r', i')$ (*cf.* définition 5.2.8 du chapitre 5).

   (b) **Aucune relation racine n'est associée avec un triplet qui fait intervenir la dernière position m :** Soit L un ensemble de triplets tel que $L = \{r''(i'', m) \in S.\theta | P_r(r'') > 1, \exists r'''(i'', m) \in \rho - S.\theta, r'' \sqsubseteq_\Omega^r r'''\}$.
   Nous allons commencer par démontrer que L ne peut pas être vide. Dans le cas où L est vide, cela impliquera que soit (i) toutes les relations de $S.\theta$ n'ont

pas de parents réguliers, ou que (ii) tous les parents réguliers de toutes les relations de $S.\theta$ sont dans $S.\theta$.

La première hypothèse est rapidement écartée puisque les relations de $S.\theta$ ne sont pas des racines et de ce fait chacune d'elles admet au moins un parent régulier.

La deuxième hypothèse est invalidée comme suit. Soit A le triplet de $S.\theta$ qui fait intervenir la position m, et qui est composé de la relation r ayant le niveau d'abstraction le plus élevé de toutes les relations associées aux triplets qui aboutissent à m. Suivant (ii), tous les parents réguliers de r sont dans $S.\theta$. Cela voudrait dire qu'il existe au moins une relation dans $S.\theta$ qui a un niveau d'abstraction plus élevé que r, ce qui est en contradiction avec notre hypothèse. D'où, L ne peut pas être vide.

Maintenant que nous avons démontré que L n'est pas vide, nous allons montrer comment que le motif S peut être généré par l'application d'une opération de suffixe sur un motif de $\mathscr{F}_{\Omega}^{k}$.

Soit S' un motif de $\mathscr{F}_{\Omega}^{k}$ tel que $S' = (S.\varsigma, (\theta \cup \{r''(i',m)\}) - \{r'(i',m)\})$, où r' et r'' sont choisis de sorte à satisfaire les conditions de la définition 5.2.9 du chapitre 5 afin que l'opération $splRelA(S', r'', r', i')$ soit valide.

Autrement dit, r'' est le plus grand parent régulier de r' qui n'est pas dans $S.\theta$ (troisième condition de la définition 5.2.9), et $r'(i',m)$ le plus grand triplet de l'ensemble L, ainsi $\Psi_{r}^{n}(r'(i',m), r''(i',m), S') = L - \{r'(i',m), r''(i',m)\}$ (définition 5.2.5 du chapitre 5), d'où la satisfaction de la deuxième condition de la définition 5.2.9. De plus, la première condition est satisfaite car aucune relation de S n'est racine. De même, les conditions de l'application de l'opération canonique $splRelC(S', r'', r', i')$ sont satisfaites. En effet, r'' est parent régulier de r', $S'.\theta$ ne contient pas $r'(i',m)$ et m est la dernière position de $S.\varsigma$.

Nous concluons que $S = splRelA(S', r'', r', i')$.

Étant donné que dans tous les cas il existe un motif de niveau k qui conduit à la

génération du motif S par l'application d'une opération de suffixe, nous concluons que l'hypothèse de départ est fausse. Ainsi, $P(k+1)$ est vraie...*(II)*

À partir des conclusions tirées de l'initialisation *(I)* et de l'induction *(II)*, nous affirmons que la propriété $P'(k)$ est vraie pour tout $k \geq 1$.

I.11 Génération non redondante d'un motif

Pour démontrer que la proposition 5.2.3 du chapitre 5, nous raisonnons par l'absurde. C'est-à-dire que nous supposons qu'un motif donné peut être généré de plusieurs façons différentes par l'application des opérations de suffixe et le développement de cette hypothèse mène à une contradiction.

Soient S_1, S_2 et S_3 trois motifs de Γ_Ω tel que $\tau(S_1) = \tau(S_2) = \tau(S_2) + 1$, et que $S_1 \sqsubseteq \Gamma_\Omega S_3$, $S_2 \sqsubseteq_{\Gamma_\Omega} S_3$, et $m = |S_3.\varsigma|$. Supposons que le motif S_3 peut être obtenu à partir des deux motifs S_1 et S_2 par l'application d'opérations de suffixe. Nous allons raisonner sur les différents cas qui peuvent se présenter. Suivant que le dernier concept de S_3 et les relations qui y sont liées, nous distinguons les situations suivantes :

1. **Aucun triplet de $S_3.\theta$ ne fait intervenir la dernière position m :** suivant la définition des différentes opérations de suffixe, seulement l'opération d'ajout ou de spécialisation de concept peut donner lieu à ce motif. Les deux cas suivants sont à distinguer.
   (a) $P_c(S_3.\varsigma[m]) = 1$, c'est-à-dire que le concept $S_3.\varsigma[m]$ est du plus haut niveau d'abstraction. Ainsi, $S_3 = addClsA(S_1, S_3.\varsigma[m])$ et $S_3 = addClsA(S_2, S_3.\varsigma[m])$. Par conséquent, $S_1.\varsigma = S_2.\varsigma$, et $S_1.\theta = S_2.\theta$. D'où, $S_1 = S_2$, ce qui est en contradiction avec l'hypothèse de départ.
   (b) $P_c(S_3.\varsigma[m]) > 1$ (le concept $S_3.\varsigma[m]$ est généralisable). Suivant la définition des opérations de suffixe, seulement l'opération de spécialisation de concept peut générer le motif S_3. Soit c le plus grand élément de l'ensemble $\{c' \in \mathscr{C}_\Omega | S_3.\varsigma[m] \sqsubseteq^r_\Omega c'\}$. Suivant la définition 5.2.7 du chapitre 5, les opérations de suffixe possibles vont porter sur la spécialisation du concept c en $S_3.\varsigma[m]$. Ainsi, $S_3 = splClsA(S_1, c, i)$ et $S_3 = splClsA(S_2, c, i)$. D'où, $S_1.\varsigma = S_2.\varsigma$, et $S_1.\theta = S_2.\theta$. Ce qui revient à dire que S_1=S_2, ce qui contredit l'hypothèse de départ.
2. **Il existe au moins un triplet de $S_3.\theta$ qui fait intervenir la dernière position m :** de la définition des opérations de suffixe, les opérations d'ajout et de spécialisation

de concept sont écartées de l'ensemble des opérations qui, appliquées à des motifs du niveau précédent, donneront lieu au motif S_3. Il reste la possibilité de générer S_3 par l'ajout ou/et la spécialisation de relations. Soient l'ensemble des relations $L = \{r'(i,m) \in S_3.\theta | P_r(r) = 1\}$, et $r(i,m)$ le plus grand élément de L. Les deux cas à distinguer sont comme suit.

(a) $r(i,m)$ **existe :** la seule opération de suffixe qui peut conduire à la génération de S_3, est l'opération d'ajout de la relation $r(i,m)$. Ainsi, $S_3 = addRelA(S_1,r,i) = addRelA(S_2,r,i)$ (*cf.* définition 5.2.8 du chapitre 5). Par conséquent, $S_1.\varsigma = S_2.\varsigma$, et $S_1.\theta = S_2.\theta$. D'où, $S_1 = S_2$, ce qui est en contradiction avec l'hypothèse de départ.

(b) $L = \emptyset$: toutes les relations qui aboutissent à la dernière position de S_3 sont d'une profondeur supérieure à 1. Ainsi, S_3 ne peut être obtenu que par l'opération de suffixe de la spécialisation de relation.

Soit l'ensemble L' tel que $L' = \{r(i,m) \in S_3.\theta | \exists r'(i,m) \in \rho - S_3.\theta, r \sqsubseteq_\Omega^r r'\}$.

Nous allons démontrer par l'absurde que S_3 sera généré par l'application d'une seule opération de suffixe.

Supposons qu'il est possible de générer S_3 par l'application d'opérations de suffixe à au moins deux motifs différents S_1 et S_2.

Soient $r''(i'',m)$ le plus grand triplet de l'ensemble L', et r''' le plus grand parent régulier de r'', tel que $r'''(i'',m)$ n'est pas dans $S_3.\theta$. D'après la définition 5.2.9 du chapitre 5, les opérations de suffixe qui génèrent S_3 ne peuvent porter que sur le remplacement du triplet $r'''(i'',m)$ par le descendant régulier $r''(i'',m)$.

Ainsi, $S_1.\varsigma = S_3.\varsigma$ et $S_1.\theta = S_3.\theta \cup \{r'''(i'',m)\} - \{r''(i'',m)\}$. De même, $S_2.\varsigma = S_3.\varsigma$ et $S_2.\theta = S_3.\theta \cup \{r'''(i'',m)\} - \{r''(i'',m)\}$. D'où, $S_1 = S_2$, ce qui est en contradiction avec l'hypothèse de départ.

De ce qui précède, nous concluons que les opérations de suffixe garantissent une génération non redondante des motifs étendus.

Annexe II

Étude de la complexité algorithmique

Soient Ω l'ontologie qui décrit le domaine considéré, $\mathscr{D}$ une base de séquences d'objets et $\mathscr{D}_\Omega$ la base des séquences d'objets étendues associée à $\mathscr{D}$. Les différentes notations que nous utilisons dans la présente étude sont résumées dans le tableau II.1.

Pour comprendre la complexité de la fouille de motifs basée sur Ω, nous estimons la taille minimale de l'espace de tous les motifs. Dans la suite, nous allons prendre en compte tous les motifs qui peuvent être générés sans aucune garantie quant à leur consistance. Ces motifs devraient être filtrés lors du test d'instanciation, et les motifs qui ne sont pas consistants ou qui ne sont pas fréquents seront rejetés.

Pour les motifs de taille k, s'ils ne sont composés que de concepts, l'espace contiendra c_m^k éléments. Ainsi, le nombre de motifs de taille allant de 1 à d_{max} est de : $\sum_{i=1}^{d_{max}} c_m^i = \frac{c_m^{d_{max}+1} - c_m}{c_m - 1}$

Avec la prise en compte des relations, pour un motif de taille k, théoriquement on aura $\frac{(k+1)*k}{2}$ couples de positions pour l'ajout de relation. Dans ce calcul, sont inclus les cas d'ajout de relation à la même position. Par exemple, pour un motif de taille 3, il existe 6 manières différentes d'ajouter une relation. De plus, pour chaque couple de positions, il existe 2^l manières différentes pour l'ajout de 1 à l relations. Ainsi, le nombre total de motifs qui pourraient être générés par l'ajout de relation à un motif de taille k est $\frac{(k+1)*k}{2} * 2^{l_m}$. Pour des motifs ayant une taille allant de 1 à d_{max}, le nombre de motifs est donné par l'expression ci-dessous :

$$\sum_{k=1}^{d_{max}} \frac{(k+1)*k}{2} * 2^{l_m} * c_m^k \backsimeq O(c_m^{d_{max}} * 2^{l_m} * d_{max}^2)$$

À titre d'exemple, dans le cas de la fouille de motifs d'objets, la taille de l'espace des motifs est donnée par $\sum_{i=1}^{d_{max}} n^i = \frac{n^{d_{max}+1} - n}{n-1} \backsimeq O(n^{d_{max}})$. Comme on peut le constater, dans

Notion	Désignation
Ontologie du domaine	Ω
Nombre de concepts de l'ontologie Ω	$c_m = \lvert\mathscr{C}_\Omega\rvert$
Nombre de relations de l'ontologie Ω	$l_m = \lvert\mathscr{R}_\Omega\rvert$
Nombre maximum de concepts/relations du plus haut niveau d'abstraction	n_h
Base de séquences d'objets étendues	$\mathscr{D}_\Omega$
Taille de $\mathscr{D}_\Omega$	d ($d = \lvert\mathscr{D}_\Omega\rvert$)
Taille maximale d'une séquence d'objets étendue (nombre d'objets)	d_{max}
Nombre d'objets distincts dans $\mathscr{D}_\Omega$	n
Nombre maximal de relations aboutissant dans un concept dans Ω	m_{relsIn}
Nombre maximal de concepts/relations prédécesseurs immédiats d'un concept (resp. d'une relation) dans Ω	$m_{predImd}$
Nombre maximal de concepts/relations successeurs immédiats d'un concept (resp. d'une relation) dans Ω	m_{Spl}
Nombre de motifs fréquents de rang maximal pouvant être générés	n_{Kp}

Tableau II.1 – Différentes notations reliées à l'étude de la complexité algorithmique de *xPMiner*.

la fouille de motifs d'objets le nombre de motifs générés dépend uniquement du nombre d'objets disponibles, et de la taille de la plus grande séquence [40]. Par contre, dans la fouille de motifs de concepts/relations, le nombre de motifs dépend à la fois du nombre de concepts, de relations, et de la taille de la plus grande séquence d'objets.

Pour ce qui est de la complexité en temps d'exécution d'une approche naïve de recherche de motifs de concepts/relations, le problème est étroitement lié au parcours de graphes. En effet, chaque motif généré sera testé avec toutes les séquences de données pour déterminer s'il est fréquent. Ce qui revient à la recherche d'un homomorphisme de chaque motif avec chaque séquence d'objets étendue. Or, le problème de test de l'homomorphisme de deux graphes est $NP - dificile$ [19].

Dans le reste de cette section, nous présentons la complexité algorithmique en nombre d'opérations et en temps d'exécution des deux algorithmes *xPMiner* et *xPMiner2*.

II.1 Complexité en nombre d'opérations

Dans la suite, nous considérons qu'une opération de base est soit l'opération canonique (cas de *xPMiner*) ou l'opération de suffixe dans le cas de l'algorithme *xPMiner2*. La limite maximale du nombre d'opérations de base nécessaires pour la génération de tous les motifs fréquents par chacun de ces deux algorithmes est donnée par la proposition 1.

Propriété 1. ***Nombre maximal d'opérations***

Soient $\mathscr{D}_\Omega$ une base de séquences d'objets étendues et k le rang maximal d'un motif fréquent, la limite du nombre maximal d'opérations nécessaires pour générer tous les motifs fréquents est donnée par :

- ***xPMiner :*** $\leq n_{Kp} * (m_{relsIn} + m_{predImd})^{(k-1)} * (m_{Spl} + m_{relsIn} + 2 * n_h)$;
- ***xPMiner2 :*** $\leq ((k-1) * n_{Kp}) * (m_{Spl} + m_{relsIn} + 2 * n_h)$.

Preuve II.1.1.

- ***xPMiner :*** *Afin d'obtenir un motif S_k du dernier niveau (rang égal à k), il faudrait explorer un chemin qui démarre du motif vide et qui arrive à S_k. Cette exploration nécessite d'effectuer un minimum de $k-1$ opérations canoniques.*

  Or, chaque motif S_t de niveau t peut être généré à partir de plusieurs motifs de niveau $t-1$. Le motif S_t peut être le fruit de la spécialisation canonique d'un maximum de $m_{predImd}$ concepts (ses prédécesseurs immédiats), et de la spécialisation canonique d'un maximum de m_{relsIn} relations. Ainsi, S_t peut avoir un maximum de $(m_{predImd} + m_{relsIn})$ prédécesseurs immédiats dans l'espace des motifs fréquents. En sommant le nombre de cas possibles du niveau 0 au niveau $t-1$, S_t nécessitera un maximum de $(m_{predImd} + m_{relsIn})^{t-1}$ opérations canoniques.

  Ainsi, la génération d'un motif de niveau k nécessitera un nombre d'opérations canoniques inférieur à $(m_{relsIn} + m_{predImd})^{(k-1)}$.

  Comme lors de la génération d'un motif fréquent, des motifs candidats sont générés, et le nombre maximal d'opérations canoniques qui les génèrent se calcule comme suit :

À partir d'un motif S_t de rang t, on peut générer un maximum de m_{Spl} motifs candidats par l'application d'opérations de suffixe pour la spécialisation de concept, n_h motifs candidats par l'ajout de concepts. De plus, en appliquant des opérations de suffixe sur les relations de S_t, on peut générer un maximum de $m_{relsIn}+n_h$ motifs de niveau $t+1$. En définitif, pour chaque motif fréquent, le nombre de candidats à générer est au maximum égal à $m_{Spl}+m_{relsIn}+2*n_h$.

Le nombre maximal des opérations canoniques nécessaires pour générer les n_{Kp} motifs de niveau k se calcule par l'expression $n_{Kp} * (m_{relsIn} + m_{predImd})^{(k-1)} * (m_{Spl}+m_{relsIn}+2*n_h)$.

- ***xPMiner2 :*** *Par l'application des opérations de suffixe d'xPMiner2, chaque motif de rang k est généré une seule fois. Ainsi, un motif de rang k est obtenu par l'application de $k-1$ opérations de suffixe en démarrant du motif vide.*

  Pour n_{Kp} motifs de niveau k, le nombre maximal de motifs des niveaux inférieurs à k qu'on fait intervenir est $(k-1)*n_{Kp}$.

  Or, la génération de chaque motif fréquent entraîne la génération d'un ensemble de motifs candidats, dont le nombre maximal est égal à celui donné dans le cas de l'algorithme xPMiner, c'est-à-dire, $m_{Spl}+m_{relsIn}+2*n_h$.

  Finalement, le nombre maximum d'opérations nécessaires pour générer n_{Kp} avec l'algorithme xPMiner2, est donné par l'expression $((k-1)*n_{Kp})*(m_{Spl}+m_{relsIn}+2*n_h)$.

II.2 Complexité en temps d'exécution

La complexité en temps d'exécution de *xPMiner* (resp. *xPMiner2*), dépend du nombre total d'opérations canoniques (resp. de suffixe) nécessaires pour la génération de tous les motifs candidats, et du coup de la procédure du test d'instanciation. La complexité algorithmique de cette dernière est étudiée dans la section suivante.

II.2.1 Complexité en temps d'exécution de la procédure de test d'instanciation

Dans la suite, une opération de base est toute opération de comparaison de deux concepts ou de deux relations.

L'algorithme d'instanciation regroupe ces appariements par préfixe, et pour un ensemble d'appariements qui ont le même préfixe, les opérations sur le préfixe seront effectuées une seule fois.

Soient s une séquence d'objets étendue et S un motif. L'algorithme d'instanciation INST() (voir l'algorithme 3 du chapitre 4) commence par une instruction *if* qui s'exécute en un temps constant, et puis appelle la procédure 5. Cette procédure est composée de deux boucles imbriquées qui cherchent les images potentielles de chaque concept de S. Étant donné que la taille maximale d'une séquence d'objets est de d_{max}, cette phase d'initialisation nécessitera au maximum d_{max}^2 opérations.

Par la suite, l'algorithme d'instanciation fait appelle à une boucle *while* pour vérifier si les images potentielles sont valides. Cette boucle est composée de deux instructions *if* qui s'exécutent en un temps constant, et d'un appel à la procédure de l'algorithme 4 de la mise en correspondance de relations. Cette dernière procédure effectue un maximum de m_{relsIn} opérations de base. Il nous reste à déterminer le nombre d'itérations de la boucle *while*. Dans le pire des cas, tous les appariements possibles seront testées. Or, pour une séquence d'objets étendue s, et un motif S, il est possible d'avoir $\frac{|s.\varsigma|!}{(|s.\varsigma|-|S.\varsigma|)!*|S.\varsigma|!}$ façons différentes de prendre $|S.\varsigma|$ éléments parmi $|s.\varsigma|$. Lors du tirage l'ordre n'est pas important, puisque les éléments d'un tirage de $|S.\varsigma|$ éléments de s peuvent être ordonnés de telle sorte à respecter leur ordre d'apparition dans s, et cet ordre est unique. Pour chacune de ces configurations, si elle est générée, des tests seront fait sur les relations. En sachant que chaque concept (resp. objet) peut être destination d'un maximum de m_{relsIn} relations différentes, un maximum de $(m_{relsIn} * m_{relsIn})$ tests de relations seront effectués au niveau de chaque concept de S avec son image potentielle dans s. Pour obtenir le nombre maximal d'opérations de comparaison de relations pour tous les concepts d'une configuration donnée, il suffit de multiplier par la taille maximale d'un motif. Ce qui

donne $|S.\varsigma|$ * m_{relsIn} * m_{relsIn}. Or, il est établi au niveau de la section 4.2.5.1 du chapitre 4 que l'algorithme d'instanciation travaille avec des préfixes et que pour un ensemble de configurations qui ont le même préfixe, les relations de ce préfixe ne sont vérifiées qu'une seule fois. Ainsi, au niveau de chaque configuration, seulement m_{relsIn} * m_{relsIn} tests de relations seront effectués. En effet, c'est seulement avec le dernier concept que ces tests seront fait.

Nous concluons que le nombre d'opérations nécessaires pour tester l'instanciation d'un motif S par une séquence d'objets étendue s est borné par :

$$\frac{|s.\varsigma|!}{(|s.\varsigma| - |S.\varsigma|)! * |S.\varsigma|!} * m^2_{relsIn}$$

II.2.2 Complexité en temps d'exécution de *xPMiner* et de *xPMiner2*

La complexité en temps d'exécution des algorithmes *xPMiner* et *xPMiner2*, peut être maintenant calculée à partir de la limite maximale du nombre d'opérations de *xPMiner* et *xPMiner2*, et de la complexité de la procédure du test d'instanciation. Ainsi, nous avons ce qui suit.

- ***xPMiner*** : $\leq n_{Kp} * 2 * (m_{relsIn} + m_{predImd})^{(k-1)} * (m_{Spl} + m_{relsIn} + 2 * n_h) * \frac{|s.\varsigma|!}{(|s.\varsigma| - |S.\varsigma|)! * |S.\varsigma|!} * m^2_{relsIn}$;
- ***xPMiner2*** : $\leq ((k-1) * n_{Kp}) * (m_{Spl} + m_{relsIn} + 2 * n_h) * \frac{|s.\varsigma|!}{(|s.\varsigma| - |S.\varsigma|)! * |S.\varsigma|!} * m^2_{relsIn}$.