Des patrons modulaires de requêtes SPARQL dans le système SWIP

Camille Pradel, Ollivier Haemmerlé, Nathalie Jane Hernandez

Des patrons modulaires de requêtes SPARQL dans le système SWIP

Camille Pradel, Ollivier Haemmerlé, Nathalie Hernandez

IRIT, Université de Toulouse le Mirail, Département de Mathématiques-Informatique, 5 allées Antonio Machado, F-31058 Toulouse Cedex 9
{camille.pradel,ollivier.haemmerle,nathalie.hernandez}
@univ-tlse2.fr

Résumé :
Le système SWIP a pour objectif l'interrogation d'entrepôts de données sémantiques par un utilisateur final. Dans cet article, nous proposons une définition de patrons de requêtes modulaires. Ces patrons sont composés de sous-patrons imbriqués, optionnels ou répétables. Ce nouveau modèle de patron est implémenté à partir d'une ontologie OWL 2. Il a été validé sur un jeu de requêtes portant sur le domaine du cinéma.

Mots-clés : Ontologie, patrons de requêtes, OWL 2, SPARQL.

1 Introduction

Les travaux menés depuis un demi-siècle sur la représentation de connaissances à l'aide de graphes étiquetés trouvent aujourd'hui une concrétisation à très grande échelle sur le Web Sémantique. Nous travaillons depuis quelques années au développement d'un environnement permettant d'interroger de manière aussi souple et aisée que possible les vastes quantités de données désormais disponibles dans les langages proposés par le W3C que sont RDF, RDFS et OWL. Le langage de requête SPARQL permet d'exprimer des requêtes dans une syntaxe "à la SQL". Il nécessite néanmoins d'exprimer des requêtes sous forme de graphes (ensembles de triplets RDF) ce qui est inenvisageable pour un utilisateur final.

Certains travaux proches des nôtres visent à générer automatiquement ou semi-automatiquement des requêtes à partir de requêtes utilisateurs exprimées sous forme de mots-clés. L'utilisateur exprime son besoin en in-

formation de manière intuitive, sans avoir à connaître le langage de requêtes ou le formalisme de représentation de connaissances utilisés par le système. Des approches ont par exemple été proposées pour générer des requêtes formelles exprimées en SPARQL (Zhou *et al.*, 2007; Tran *et al.*, 2009). Dans ces systèmes, la génération de la requête passe notamment par l'appariement des mots-clés aux entités sémantiques de la base de connaissances puis par la construction de graphes requêtes établissant des liens entre les entités appariées.

Notre approche diffère des approches existantes sur deux points essentiels. Tout d'abord, comme nous l'avons présenté dans (Pradel *et al.*, 2011), nous avons dépassé les requêtes exprimées sous forme de simples mots-clés en permettant désormais l'expression de relations dans les requêtes – notre ambition étant d'atteindre à terme un langage de requêtes aussi proche que possible de la langue naturelle. Ensuite, le mécanisme de construction de requêtes que nous proposons se fonde sur l'utilisation de patrons, qui sont des modèles de requêtes que nous adaptons aux besoins exprimés par l'utilisateur. En effet, dans le cadre d'applications réelles, les requêtes soumises par les utilisateurs sont généralement des variations autour de quelques familles de requêtes typiques. L'utilisation de patrons nous affranchit notamment de la phase d'exploration de l'ontologie en vue de lier les concepts identifiés à partir des mots-clés, puisque les relations potentiellement pertinentes apparaissent dans les patrons. Le processus bénéficie donc des familles de requêtes pertinentes pré-établies pour lesquelles nous savons qu'un réel besoin en information existe.

Le système SWIP [1] reçoit en entrée une ontologie OWL du domaine considéré ainsi qu'une requête exprimée dans un langage que nous qualifions de langage pivot, car il est dans notre dispositif le pivot entre la langue naturelle et SPARQL. Dans la première étape du traitement, la requête est appariée à des éléments de notre base de connaissances (concepts, propriétés, instances). Nous associons alors les patrons de requêtes à ces éléments. Les différentes associations sont proposées à l'utilisateur sous forme de phrases descriptives exprimées en langue naturelle. L'utilisateur sélectionne la phrase correspondant à la requête qu'il souhaite effectivement poser. SWIP peut finalement construire la requête SPARQL voulue, en adaptant le patron de requête considéré.

Dans cet article, nous approfondissons notre travail dans deux directions. Nous enrichissons dans un premier temps notre modèle de patrons. En effet, nous avons ressenti le besoin de proposer des patrons modulaires,

1. Semantic Web Interface using Patterns

dont certaines parties peuvent être optionnelles ou répétables. L'objectif de cette évolution est de diminuer le nombre de patrons nécessaires à la couverture d'un domaine en les factorisant, ainsi que de réduire le silence auquel on peut être confronté en exécutant nos requêtes. Dans un second temps, nous proposons une ontologie permettant de représenter dans les langages du Web Sémantique les patrons eux-mêmes, à l'image de ce qui est fait dans (Dietrich & Elgar, 2005) pour les patrons de conception issus du génie logiciel. Outre le fait d'homogénéiser les connaissances que nous manipulons, cela nous permet de partager, de comparer, de différencier et de faire évoluer les patrons qui deviennent eux-mêmes interrogeables au moyen de requêtes SPARQL. Nous pouvons ainsi envisager la constitution d'entrepôts de patrons, ces entrepôts pouvant être répartis sur le Web.

La partie 2 de cet article présente une nouvelle définition des patrons de requêtes SPARQL. La section 3 présente la sémantique associée à ces patrons en montrant comment ils peuvent être instanciés en requêtes SPARQL. La section 4 présente l'ontologie utilisée par notre système pour représenter des patrons en RDF. Nous terminons l'article par une présentation de l'implémentation et des résultats obtenus.

2 Les patrons de requêtes

Nous avons introduit la notion de patron et nous en avons proposé une première définition dans (Pradel *et al.*, 2011). Rappelons qu'un patron est composé d'un graphe RDF qui est le prototype d'une famille de requêtes typiques. Ce patron est caractérisé par un sous-ensemble des sommets et arcs de ce graphe – représentant aussi bien des concepts que des propriétés ou des types de données – que nous appelons sommets et arcs "qualifiants", et qui peuvent être modifiés pendant la construction de la requête finale. Le patron est également décrit par une phrase en langue naturelle dans laquelle une sous-chaîne distincte est associée à chacun des sommets qualifiants. Cette phrase descriptive a également vocation à être modifiée de façon à ce qu'elle corresponde exactement à la requête SPARQL qui sera finalement générée. Rappelons également que, dans l'état actuel des choses, les patrons sont conçus "manuellement" par des experts qui connaissent le domaine d'application et la forme habituelle des graphes de requêtes.

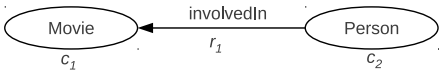
2.1 Limite des anciens patrons

L'expérience acquise au cours de ces dernières années dans le développement de patrons sur différents domaines applicatifs nous a amenés

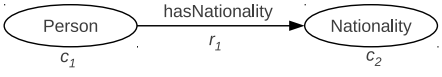
à proposer des patrons modulaires, dont certaines parties peuvent être optionnelles, et d'autres peuvent être répétées. Supposons que nous souhaitions autoriser la génération de requêtes concernant :

- les personnes impliquées dans un film : le patron de la figure 1(a) permet de répondre, par spécialisation de la propriété qualifiante *involvedIn*, notamment aux requêtes "Which actors play in *Biutiful* ?" et "Which movies where directed by Alejandro González Inárritu ?" ;
- la nationalité des personnes : le patron de la figure 1(b) permet de répondre par exemple à "What is the nationality of Joel Coen ?" ;
- les films, leurs genres et leurs dates de sortie : le patron de la figure 1(c) permet de répondre par exemple à "Which suspense movies were released in 2008 ?"

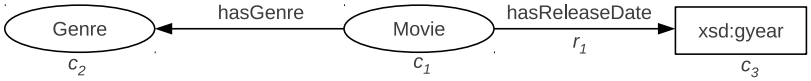
Dans les exemples de l'article, les sommets et arcs qualifiants sont repérés par un identificateur. D'autre part, chaque sous-chaîne du template de phrase descriptive est soulignée et indicée par l'identificateur de l'élément qualifiant avec lequel elle est mise en bijection.



(a) A person$_{c_2}$ which was involved in$_{r_1}$ a movie$_{c_1}$



(b) A person$_{c_2}$ which is of unknown nationality$_{c_3}$

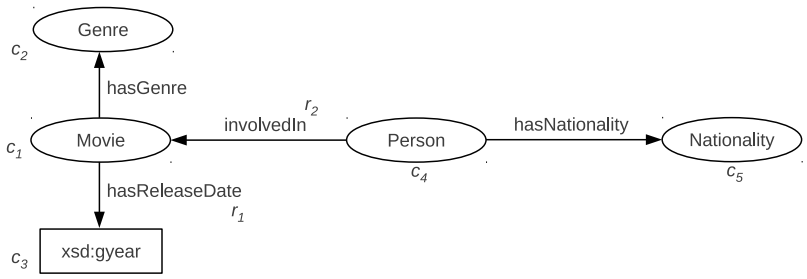


(c) A movie$_{c_1}$ of some genre$_{c_2}$ was released on some date$_{c_3}$

FIGURE 1 – Exemples illustrant les défauts des anciens patrons.

Afin d'éviter la multiplication de patrons très spécifiques, il pourrait être judicieux de regrouper ces trois patrons en un seul, comme nous le proposons en Figure 2.

L'inconvénient qu'induit une telle factorisation est qu'elle peut entraîner une baisse du rappel. Imaginons par exemple que la base de patrons ne soit composée que du patron présenté en Figure 2, et que l'utilisateur pose comme requête "Which movies were released in 2008 ?". La requête générée par le système SWIP conduirait à la phrase descriptive : "A person which is of unknown nationality was involved in a movie of some genre which was released in 2008". Outre le fait que cette phrase descriptive présente une partie inutile et dégrade l'ergonomie du système en rendant



A person$_{c_4}$ which is of unknown nationality$_{c_5}$ was involved in$_{r_1}$ a movie$_{c_1}$ of some genre which was released in some year$_{c_3}$

FIGURE 2 – Patron réunissant les trois patrons de la Figure 1.

peu compréhensible la phrase par rapport à la requête initiale, la requête SPARQL générée retournera tous les films sortis en 2008 (comme demandé) à condition que ces films aient au moins un genre et un réalisateur connu d'une nationalité connue ; autrement dit, seront ignorés les films dont la base de connaissance interrogée ne mentionne pas le genre, la nationalité du réalisateur ou encore le réalisateur lui-même.

2.2 Évolution de la définition d'un patron

Nous proposons donc une nouvelle définition des patrons afin de les rendre modulaires. Il est désormais possible de définir dans chaque patron des parties optionnelles (qui pourront être omises dans la requête SPARQL), ou répétables (qui pourront apparaître plusieurs fois dans la requête SPARQL). Chacune de ces parties de patrons – que nous appelons "sous-patrons" – est caractérisée par une cardinalité minimum et une cardinalité maximum. Une cardinalité minimum de 0 signifie que le sous-patron est optionnel, une cardinalité maximum supérieure à 1 signifie que le sous-patron est répétable. L'imbrication de sous-patrons est possible.

Soit G un graphe et v un sommet de ce graphe ; on note $G \setminus v$ le graphe G privé du sommet v et de tous les arcs incidents à ce sommet.

Définition 1
Un patron p est un quadruplet $(G, \mathcal{Q}, \mathcal{SP}, \mathcal{S})$ tel que :

- *G est un graphe RDF connexe qui décrit la structure générale du patron et représente une famille de requêtes. Ce graphe ne doit contenir que des triplets répondant à la structure présentée en Figure 3 ;*
- *$\mathcal{Q}$ est un sous-ensemble d'éléments de G, appelés éléments qualifiants ; ces éléments sont considérés comme caractéristiques du pa-*

tron et seront pris en compte lors de l'opération d'association de la requête utilisateur au patron concerné. Un élément qualifiant est soit un sommet (classe ou type de données), soit un arc (propriété d'objet ou propriété de données) de G ;

- *$\mathcal{SP}$ est l'ensemble des sous-patrons sp de p tel que, $\forall sp = (SG, v, card_{min}, card_{max}) \in \mathcal{SP}$, on a :*
  - *SG est un sous-graphe de G et v un sommet de SG (et donc de G), tels que $G \setminus v$ est non connexe (v est un sommet d'articulation de G que nous appelons* sommet de jonction*) et admet $SG \setminus v$ comme composante connexe (i.e. tous les sommets de cette composante connexe appartiennent au graphe du sous-patron) ;*
  - *au moins un sommet ou un arc de SG est qualifiant ;*
  - *$card_{min}, card_{max} \in \mathbb{N}$ et $0 \leq card_{min} \leq card_{max}$; ce sont les cardinalités respectivement minimale et maximale de sp qui permettent de définir les caractères optionnel et répétable de sp.*
- *$\mathcal{S} = (s, (sw_1, sw_2, ...sw_n), (w_1, w_2, ...w_m))$ est un template de phrase descriptive dans laquelle n sous-chaînes sw_i correspondent aux n sous-patrons du patron, m sous-chaînes distinctes w_j correspondent aux m éléments qualifiants du patron.*

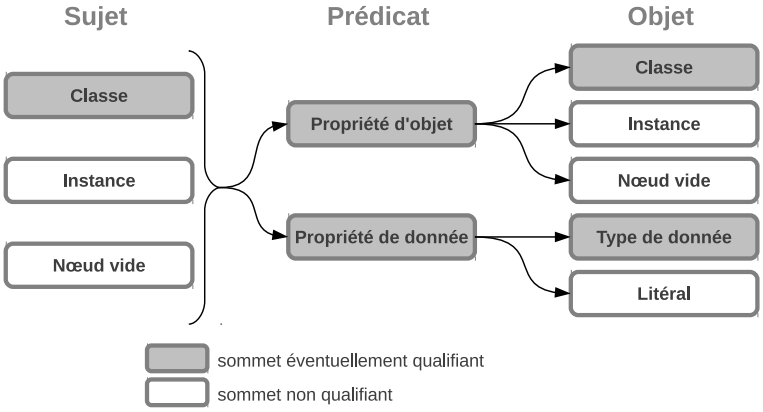
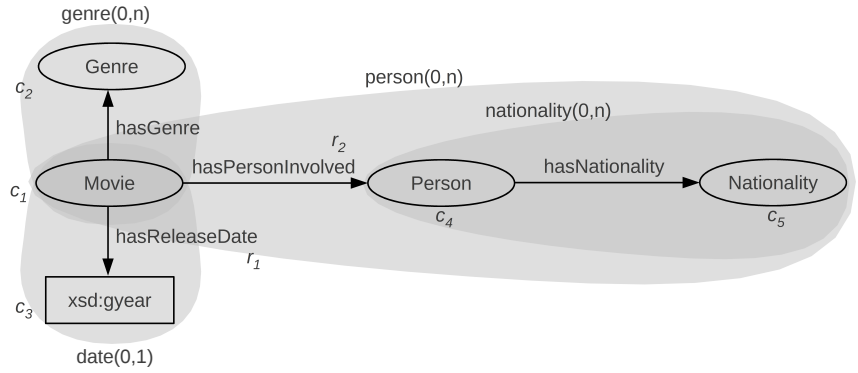


FIGURE 3 – Contraintes sur les triplets constituant les graphes patrons.

Notons que la seule contrainte morphologique d'un sous-patron est qu'il doit être relié au reste du graphe requête par un seul sommet, ce qui fait de ce sommet un sommet d'articulation du graphe requête. Cela nous garantit que si le sous-patron est optionnel, la suppression des sommets de ce sous-patron à l'exception du sommet d'articulation de la requête finale ne pourra

pas entraîner une déconnexion du graphe requête. D'autre part, si ce sous-patron est répété, il sera entièrement dupliqué à l'exception du sommet d'articulation comme nous l'illustrons maintenant au moyen d'un exemple.

La Figure 4 propose une évolution du patron présenté en Figure 2 se conformant à la nouvelle définition d'un patron. Ce patron se compose de quatre sous-patrons que nous avons appelés *genre*, *date*, *person* et *nationality*. Tous ces sous-patrons sont optionnels. Ils sont également répétables, à l'exception du sous-patron *date* : on considère qu'un film ne peut avoir qu'une date de sortie. Dans le template de phrase descriptive, les parties de phrase correspondant à un sous-patron sont représentées entre crochets et indicées par l'identificateur du sous-patron, les sommets qualifiants sont soulignés et indicés par l'élément qualifiant concerné.



A movie$_{c_1}$ [of genre $\underline{c_2}]_{genre}$ [that was released on $\underline{c_3}]_{date}$
$\big[$has for person involved$_{r_2}$ a person$_{c_4}$ [which is $\underline{c_5}]_{nationality}\big]_{person}$

FIGURE 4 – Évolution du patron 2.

Si l'on considère le sous-patron *nationality* et le sous-graphe associé $SG_{nationality}$, le sommet de jonction de ce sous-patron est identifié c_4 dans le schéma. En effet, conformément à la définition 1, il est le seul sommet dont la suppression entraîne la déconnexion du graphe du patron avec pour composante connexe le graphe du sous-patron privé de c_4. Si l'on génère une requête à partir de ce patron dans laquelle le sous-patron est omis, le graphe de la requête SPARQL sera amputé de tous les sommets de $SG_{nationality}$ – et des arcs induits – à l'exception du sommet de jonction c_4 ; la phrase descriptive sera elle aussi amputée de la partie faisant référence au sous-patron omis. Si au contraire le sous-patron est répété,

le graphe de la requête SPARQL contiendra autant de sous-graphes distincts isomorphes à $SG_{nationality}$ à l'exception du sommet de jonction c_4 qui n'apparaîtra qu'une seule fois et sera commun à tous les sous-graphes en question ; dans la phrase descriptive, la partie relative au sous-patron répété sera elle aussi répétée afin de représenter chacun des sous-graphes issus du sous-patron.

3 Sémantique des patrons

Nous allons maintenant étudier la façon dont s'instancient les patrons de requêtes, autrement dit comment on passe d'un patron de requête à une requête SPARQL effective. Nous allons ainsi pouvoir caractériser la sémantique associée aux patrons en étudiant les différentes façons d'instancier les éléments de patrons, puis les sous-patrons, et enfin les patrons eux-mêmes.

3.1 Instanciation d'élément de patron

Nous commençons par étudier l'instanciation d'un élément qualifiant d'un patron. En d'autres termes, nous allons voir comment est transformé le graphe requête lorsqu'un de ses éléments qualifiants est rapproché d'un élément de la requête utilisateur.

Soit un patron $p = (G, \mathcal{Q}, \mathcal{SP}, \mathcal{S})$. Pour tout $q \in \mathcal{Q}$ élément qualifiant de p et pour toute ressource α extraite de la requête utilisateur (qui peut être une classe, une instance ou une propriété), on note $I(p, q \leftarrow \alpha) = (G', \mathcal{Q}', \mathcal{SP}', \mathcal{S}')$ le patron obtenu après instanciation de q par la ressource α dans le patron p. Cette instanciation n'est possible que si q et α sont compatibles, comme expliqué dans la description des étapes d'appariement et d'association dans (Pradel *et al.*, 2011). Sans les décrire formellement, nous rappelons les cas de compatibilité, et décrivons l'instanciation de l'élément qualifiant correspondant en l'illustrant par un exemple construit à partir du patron de la Figure 4.

1. q est une classe et α une instance de q ou d'une de ses sous-classes. Dans ce cas-là, l'instanciation du sommet qualifiant consiste à remplacer l'URI de la classe par l'URI de l'instance. Par exemple dans la Figure 5, le sommet c_4 Person a été instancié en Joel Coen.
2. q est un type de données et α une valeur correspondant au type q. Dans ce cas-là, l'instanciation du sommet qualifiant consiste à rem-

placer l'URI de la classe par la valeur α. Par exemple dans la Figure 5, le sommet c_3 Date (xsd :gyear) a été instancié en 2008.

3. q est une propriété et α la même propriété ou une de ses sous-propriétés. Dans ce cas-là, l'instanciation de l'arc qualifiant consiste à remplacer l'URI de l'arc par l'URI de la propriété α. Par exemple dans la Figure 5, l'arc r_2 hasPersonInvolved a été instancié en hasDirector.
4. q est une classe et α la même classe ou une de ses sous-classes. Dans ce cas-là, G' est le graphe G dans lequel q a été remplacé par un nœud vide et augmenté d'un triplet supplémentaire indiquant que ce nœud vide est de type α. Par exemple dans la Figure 5, le sommet c_1 est remplacé par un nœud vide typé par la classe *short film*, sous-classe de *movie*.
5. Enfin, un élément de patron peut également être instancié s'il n'a été associé à aucun élément de la base de connaissances ($\alpha = \varnothing$). Le résultat de cette instanciation est tel que :
   – si q est une propriété, alors $G' = G$;
   – si q est une classe ou un type de données, alors G' est le graphe G dans lequel q a été remplacé par un nœud vide et augmenté d'un triplet supplémentaire indiquant que ce nœud vide est de type q.

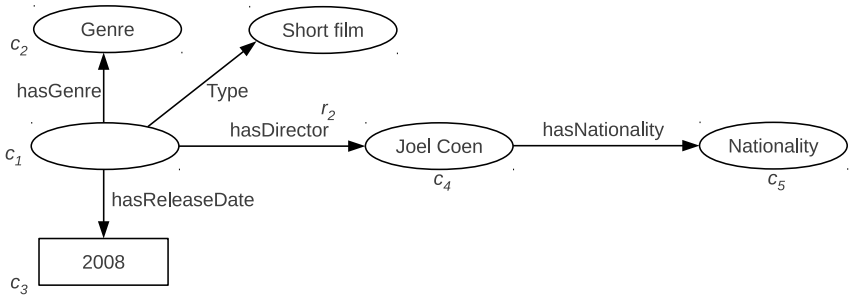


FIGURE 5 – Instanciation partielle d'un graphe patron, selon l'exemple déroulé précédemment. Il ne s'agit pas encore d'un graphe requête.

3.2 Instanciation de sous-patron

Les sous-patrons pouvant être imbriqués, nous définissons ici leur instanciation de manière récursive. Soit un patron $p = (G, \mathcal{Q}, \mathcal{SP}, s)$ et $sp = (SG, s, card_{min}, card_{max}) \in \mathcal{SP}$ un sous-patron de p. Le patron p' est issu

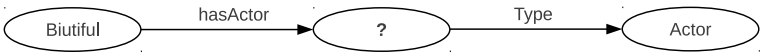
de l'instanciation de sp dans p si p' a pour graphe G' qui est le graphe G dans lequel SG' peut apparaître n fois ($card_{min} \leq n \leq card_{max}$) avec pour chaque occurrence de SG' une combinaison d'instanciations d'éléments qualifiants différente. SG' est obtenu en appliquant à SG l'instanciation de l'ensemble des sous-patrons qu'il contient puis l'instanciation successive de l'ensemble des éléments qualifiants restants,

3.3 Instanciation de patron

Un patron peut être considéré comme un sous-patron n'étant inclus dans aucun autre, et dont les cardinalités minimale et maximale sont égales à 1. Le mécanisme d'intanciation reste donc le même. Dans le résultat obtenu, seuls le graphe requête G' et la phrase descriptive s' sont utiles pour la suite, les ensembles $\mathcal{Q}'$ des sommets qualifiants et $\mathcal{SP}'$ des sous-patrons étant de toute façon vides.

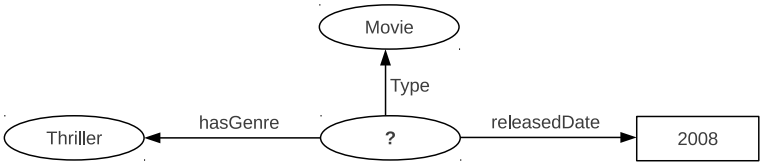
La famille de requêtes représentée par un patron est l'ensemble des requêtes qui peuvent être obtenues par instanciation de ce patron.

Nous présentons successivement dans les Figures 6, 7, 8 et 9 les graphes requêtes générés par le système SWIP pour les quatre requêtes suivantes (toutes les requêtes sont des instanciations du patron 4) : (i) "Which actors play in the movie Biutiful ?" ; (ii) "Which thrillers were released in 2008 ?" ; (iii) "Which movies were realised by a French director ?" ; (iv) "Which movies were directed by Coen brothers ?".



Phrase descriptive : Biutiful has for actor **some actor**.

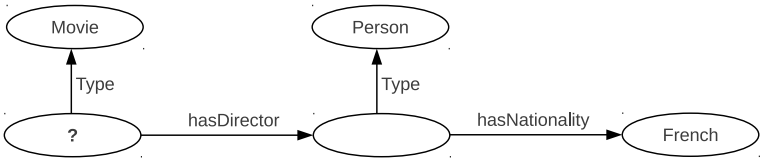
FIGURE 6 – Instanciation du patron 4 correspondant à la requête (i).



Phrase descriptive : **A movie** of genre thriller was released in 2008.

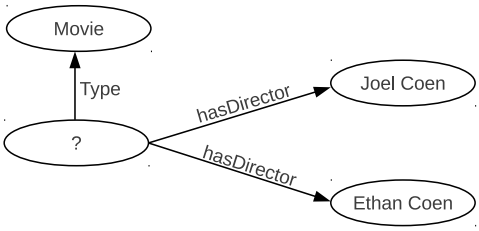
FIGURE 7 – Instanciation du patron 4 correspondant à la requête (ii).

Remarquons qu'il était auparavant impossible d'interpréter la requête (iv), car "Coen brothers" n'était apparié qu'à un seul élément de la base



Phrase descriptive : **A movie** that was directed by some person which is French.

FIGURE 8 – Instanciation du patron 4 correspondant à la requête (iii).



Phrase descriptive : **A movie** that was directed by Joen Coen and was directed by Ethan Coen.

FIGURE 9 – Instanciation du patron 4 correspondant à la requête (iv).

de connaissances, “Joel Coen” ou “Ethan Coen”. La répétabilité du sous-patron concerné permet désormais de considérer les deux appariements dans une même instanciation et d’obtenir ainsi la requête escomptée.

4 Une ontologie de patrons de requêtes

Dans cette section, nous présentons brièvement l’ontologie que nous avons construite pour modéliser les patrons et faciliter ainsi la gestion, le partage et l’évolution de ces patrons.

4.1 Qualité et bonnes pratiques

Durant la conception de cette ontologie, nous nous sommes efforcés d’appliquer les techniques couramment recommandées dans la communauté. Nous avons notamment implémenté des patrons de conception d’ontologies. Le plus important de ces patrons est le patron de conception *normalization* introduit dans (Rector, 2003) ; cette méthode permet de construire des ontologies modulaires et réutilisables en définissant les classes par les propriétés que doivent vérifier leurs instances ; le concepteur de l’ontologie n’a ainsi pas à se soucier des propriétés de subsomption : la

taxonomie est inférée automatiquement par le moteur d'inférences (une classe *A* subsume une classe *B* si les instances de *B* présentent au moins toutes les propriétés des instances de *A*).

Parmi les autres bonnes pratiques utilisées, citons l'alignement de notre ontologie avec des ontologies de référence (comme FOAF présenté dans (Brickley, 2007)) et la définition pour chaque propriété d'objet d'une propriété inverse facilitant ainsi les manipulations.

Enfin, l'ontologie ayant été décrite en OWL2, nous avons tiré parti des progrès de ce langage en terme d'expressivité présentés dans (OWL 2 new features, 2009). Nous avons notamment défini des chaînes de propriétés permettant de plus grandes capacités d'inférence sans avoir recours à un langage dédié à l'expression de règles, et utilisé la possibilité de considérer une ressource avec deux visions différentes (*punning* présenté en 4.2.2).

4.2 Description de l'ontologie

Nous utilisons dans la suite les préfixes usuels (`rdf`, `rdfs`, `xsd` et `owl`) ; nous définissons également un nouveau préfixe, spécifique aux entités déclarées dans notre ontologie : au nom de préfixe `swip`, nous associons l'IRI `http://swip.alwaysdata.net/ontologies/SwipOntology`.

Notre ontologie est construite selon les principes du patron de conception *normalization* introduit plus haut ; les classes sont donc définies à l'aide des propriétés. C'est pourquoi nous présentons dans un premier temps les propriétés, puis les classes principales de l'ontologie.

4.2.1 Les propriétés

La Figure 10 représente la hiérarchie de propriétés de notre ontologie, caractérisées par leurs domaines et co-domaines. À chaque propriété correspond une propriété inverse.

La propriété `swip:makesUp` est une relation de méronymie générique ; elle est spécialisée par différentes sous-propriétés choisies en fonction de la nature de la partie et du tout comme `swip:isSubjectOf`, `swip:isPropertyOf`, `swip:isObjectOf`, `swip:isSentenceOf` et `swip:isSubsentenceOf`.

`swip:isSentenceOf` et `swip:isSubsentenceOf` permettent d'associer le template de phrase descriptive à un patron. La propriété `swip:targets` exprime la relation entre un élément de patron et l'élément (classe, propriété ou type de données) de l'ontologie cible auquel cet élément de patron réfère ; elle permet ainsi de spécifier les possibilités d'appariement.

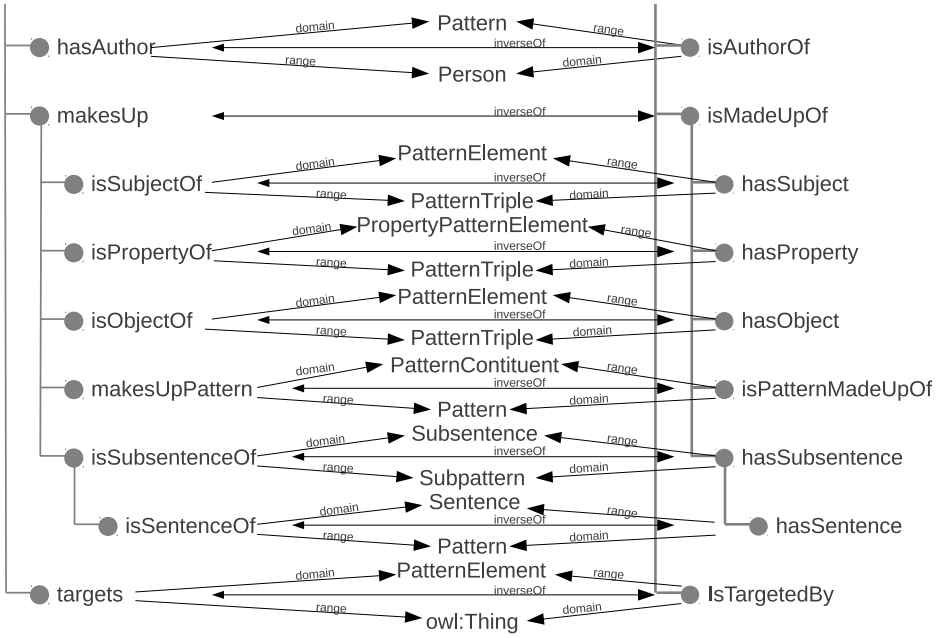


FIGURE 10 – Les propriétés d'objet de l'ontologie SWIP.

Les propriétés de données `swip:hasCardinalityMin` et `swip:hasCardinalityMax` permettent de spécifier les cardinalités des sous-patrons.

4.2.2 Les classes

La Figure 11 présente la hiérarchie des classes principales de notre ontologie SWIP, également reliées par la relation de méronymie.

Les sous-patrons sont représentés par la classe `swip:Subpattern`, qui subsume les classes distinctes `swip:PatternTriple` et `swip:SubpatternCollection`.

Une instance de la classe `swip:PatternTriple` est un triplet du graphe d'un patron. Un triplet est caractérisé par son sujet (relation `swip:hasSubject`), sa propriété (relation `swip:hasProperty`) et son objet (relation `swip:hasObject`). Les valeurs des ces propriétés sont des instances de la classe `swip:PatternElement`, qui sont les composants atomiques du patron. Un élément de patron fait référence à un type de littéral (`swip:LiteralPatternElement`), une classe (`swip:ClassPatternElement`) ou une propriété (`swip:PropertyPatternElement`) de l'ontologie. Par exemple, une instance de `swip:ClassPatternElement` im-

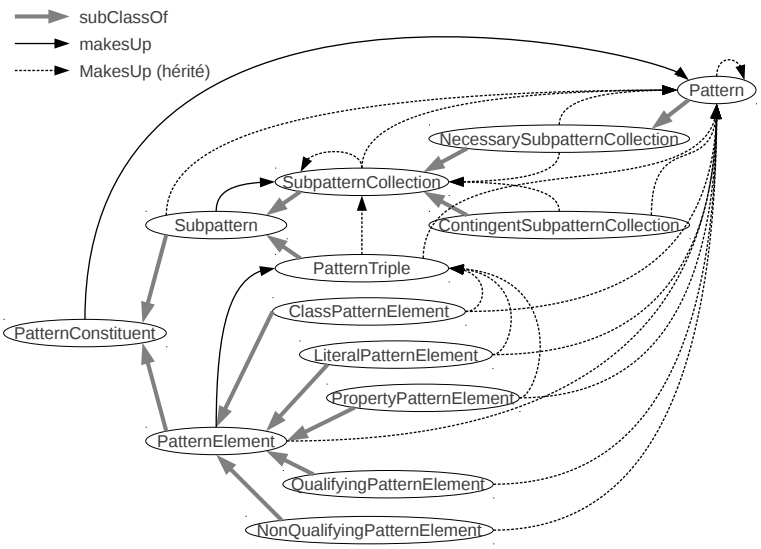


FIGURE 11 – Les principales classes permettant de décrire des patrons, les relations de taxonomie qui les hierarchisent, ainsi que les relations de méronymie possibles entre les instances de ces classes.

pliquée dans un patron sera reliée par la relation `swip:targetKBElement` à une classe de l'ontologie cible ; cette classe sera alors dans ce contexte, considérée comme une instance. Cette méthode, appelée *punning*, est autorisée en OWL 2. Cependant son utilisation est impossible à modéliser dans une ontologie OWL 2, car elle nécessiterait l'utilisation d'une partie du vocabulaire réservé (`owl:Class`, `owl:Property`, `owl:Literal`). `owl:Thing` est utilisé en lieu et place de chacune de ces entités proscrites, ce qui ne permet pas de différencier les trois sous-classes de `swip:PatternElement` autrement qu'en les déclarant distinctes.

5 Implémentation et validation

Une grammaire BNF permettant d'exprimer les patrons et l'ensemble de leurs composants a été définie. Lors de leur conception, les patrons sont représentés dans un fichier texte qui respecte cette grammaire. Ces patrons au format texte sont alors automatiquement traduits en RDF, en s'appuyant sur l'ontologie présentée plus haut. Cette méthode s'avère bien plus pratique que de définir directement en RDF les instances des classes de l'ontologie et les relations qui les unissent, et nous semble être une alter-

native viable à l'utilisation d'une interface dédiée à l'édition et l'évolution de patrons, dont le développement serait bien plus coûteux.

La validité des patrons générés par rapport au modèle défini dans l'ontologie peut être vérifiée à l'aide du validateur de contraintes d'intégrité de Pellet, présenté dans (Tao *et al.*, 2010). Cet outil, en contradiction avec les exigences portant sur le comportement d'un agent du Web sémantique, fait l'hypothèse du monde fermé (*closed world assumption*) et de l'identifiant unique (*unique name assumption*), et permet ainsi de considérer OWL comme un langage de validation pour RDF et donc de valider la conformité d'un ensemble de données RDF à une ontologie OWL.

Enfin, pour valider le nouveau modèle de patron proposé, nous avons redéfini les patrons construits lors des expérimentations présentées dans (Pradel *et al.*, 2011). Ces travaux portaient sur 160 requêtes sur le thème du cinéma recueillies auprès de 24 personnes différentes. Nous avions alors obtenu 12 patrons. En utilisant le modèle de patrons modulaires, nous obtenons aujourd'hui 4 patrons. Les patrons sont certes plus grands et plus complexes mais leur faible nombre rend leur manipulation plus aisée et le découpage en sous-patrons se fait de manière naturelle.

6 Conclusion et perspectives

Nous avons proposé une définition plus modulaire des patrons de requêtes du système SWIP au travers d'une ontologie permettant de modéliser ces patrons. Les patrons sont désormais composés de sous-patrons imbriqués et présentant une information de cardinalité pouvant les rendre optionnels ou répétables. Cette démarche présente plusieurs avantages :

- chaque patron couvre un plus grand nombre de requêtes ; le nombre de patrons nécessaires pour couvrir un domaine est donc plus faible,
- les patrons peuvent être publiquement accessibles via les moyens du Web sémantique : regroupés dans des entrepôts RDF, ils peuvent être obtenus via le protocole et langage d'interrogation SPARQL, et sont conformes à un modèle défini dans une ontologie OWL2,
- la traduction de cette ontologie en un modèle orienté objet et donc l'exploitation de patrons de requêtes dans une application est permise par des méthodes fiables permettant une analogie entre ontologie et modèle de données orienté objet, comme celle présentée dans (Evermann & Wand, 2005),
- la mise en place d'entrepôts regroupant des patrons pourrait être une bonne base pour la phase de construction des patrons : des experts

pourraient construire ces patrons de manière collaborative via une interface graphique.

Nous commençons maintenant une nouvelle phase de ce travail : l'étude de la génération automatique de patrons de requêtes ; des requêtes réelles seront traduites en patrons très spécifiques, puis des itérations successives les feront évoluer vers des patrons de plus en plus génériques.

Références

BRICKLEY D. (2007). Foaf vocabulary specification 0.9. *http ://xmlns. com/foaf/spec/20070524. html*.

DIETRICH J. & ELGAR C. (2005). A formal description of design patterns using owl. In *Proceedings of the 2005 Australian conference on Software Engineering*, p. 243–250, Washington, DC, USA : IEEE Computer Society.

EVERMANN J. & WAND Y. (2005). Ontology based object-oriented domain modelling : fundamental concepts. *Requirements Engineering*, **10**(2), 146–160.

OWL 2 NEW FEATURES (2009). Owl 2 web ontology language new features and rationale. Web site. http ://www.w3.org/TR/2009/REC-owl2-new-features-20091027/.

PRADEL C., HAEMMERLÉ O. & HERNANDEZ N. (2011). Expression de requetes SPARQL a partir de patrons : prise en compte des relations (regular paper). In A. MILLE, Ed., *Journees Francophones d'Ingénierie des Connaissances (IC), Chambery, 16/05/11-20/05/11*, p. 771–786 : Presses Universitaires des Antilles et de la Guyane.

RECTOR A. (2003). Modularisation of domain ontologies implemented in description logics and related formalisms including owl. In *Proceedings of the 2nd international conference on Knowledge capture*, p. 121–128 : ACM.

TAO J., SIRIN E., BAO J. & MCGUINNESS D. (2010). Integrity constraints in owl.

TRAN T., WANG H., RUDOLPH S. & CIMIANO P. (2009). Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In *ICDE*, p. 405–416 : IEEE.

ZHOU Q., WANG C., XIONG M., WANG H. & YU Y. (2007). Spark : Adapting keyword query to semantic search. In *ISWC/ASWC*, p. 694–707.