Une extension pour RDF/RDFS utilisant des relations procédurales

Jean-François Baget*

* *INRIA Sophia-Antipolis & LIRMM(CNRS - UM2)*
LIRMM, 161 rue Ada, 34392 Montpellier Cedex 5
baget@lirmm.fr

RÉSUMÉ. *Nous présentons ici une extension du langage RDF/S utilisé dans le cadre du web sémantique qui permet de représenter comme des objets de première classe des contraintes fonctionnelles portant sur des littéraux typés (qui sont des contraintes externes au graphe dans le langage de requêtes SPARQL). Nous présentons deux sémantiques pour ces nouveaux objets, la deuxième affaiblissant celle de RDF/S. Avec cette deuxième sémantique, répondre à une requête, avec certaines restrictions syntaxiques sur le graphe cible, reste un problème NP-complet.*

ABSTRACT. *We present here an extension of the semantic web RDF/S language, allowing to represent functional constraints on litterals as first-class objects (instead of being external constraints in the SPARQL query language). We propose two semantics for these new objects, the second one weakening the usual RDF/S semantics. With this latter semantics, query answering using this new language, considering syntactic restrictions on the database graph, remains an NP-complete problem.*

MOTS-CLÉS : *Web sémantique, langages, raisonnement, RDF/S, types de données, relations procédurales, règles d'inférence.*

KEYWORDS: *Semantic web, languages, reasonings, RDF/S, datatypes, procedural relations, inference rules.*

1. Motivation

RDF (Resource Description Framework) et son extension RDFS (RDF Schema) sont deux langages de représentation de connaissances développés au sein du W3C pour annoter des ressources dans le cadre du web sémantique. Un *graphe RDF* est défini par un ensemble de triplets `(sujet, propriété, objet)`qui sont chacun représentés dans un graphe par un arc (étiqueté `propriété`) allant du sommet identifié par son étiquette `sujet` à celui étiqueté `objet`. Ces étiquettes seront soit des *urirefs* (utilisées comme constantes dans le web) soit des variables appelées *blancs* (elles ne sont pas dessinées dans le graphe).

Les graphes RDF sont munis d'une sémantique en théorie des modèles (Hayes, 2004) : une *interprétation* est un triplet constitué d'un *domaine* Δ, d'une *fonction d'interprétation* ι associant à chaque uriref un élément du domaine, et d'une application partielle ϵ associant à certains éléments du domaine (ceux qui interprètent des propriétés) un ensemble de paires d'éléments du domaine (qui forment *l'extension* de la propriété). L'interprétation d'un vocabulaire composé d'urirefs permet de donner une valeur de vérité à un graphe RDF : on dit qu'une interprétation $(\Delta, \iota, \epsilon)$ est un modèle pour un graphe RDF G si et seulement si il existe une application ι' des *termes* de G (les urirefs et les blancs qui apparaissent dans au moins un triplet de G) dans le domaine Δ qui préserve les constantes (si u est une uriref, alors $\iota'(u) = \iota(u)$) et la structure (si (s, p, o) est un triplet de G, alors $(\iota'(s), \iota'(o)) \in \epsilon(\iota'(p))$). Cette définition de la sémantique d'un graphe RDF nous permet de définir les notions de *satisfiabilité* d'un graphe(il existe un modèle), de *validité* d'un graphe (toutes les interprétations sont des modèles), ainsi que de *conséquence sémantique* entre graphes (on dit que Q est conséquence sémantique de G, et on note $Q \models G$ lorsque tous les modèles de G sont des modèles de Q).

(Gutiérrez *et al.*, 2004; Baget, 2005) caractérisent la conséquence sémantique entre graphes par une application entre les termes de ces graphes : Q est conséquence sémantique de G si et seulement si il existe une application (appelée *homomorphisme*) π des termes de Q dans les termes de G qui préserve les constantes (si u est une uriref de Q, alors $\pi(u) = u$) ainsi que la structure (si (s, p, o) est un triplet de Q, alors $(\pi(s), \pi(p), \pi(o))$ est un triplet de G). Ainsi, décider si un graphe RDF est conséquence sémantique d'un autre est un problème NP-complet. Vouloir calculer l'ensemble des réponses à un graphe Q dans un graphe G consiste à calculer l'ensemble des homomorphismes de Q dans G. Cette propriété est utilisée pour définir le langage de requêtes SPARQL (Prudhommeaux *et al.*, 2006), dédié à l'interrogation de bases de connaissances RDF/S : la forme la plus simple d'une requête SPARQL est SELECT $\vec{u}$ FROM G WHERE Q où G est le graphe RDF cible, Q est le "graph pattern" écrit en RDF, encodant la structure de la requête, et $\vec{u}$ est un tuple de blancs de Q. Les réponses à la requête sont alors les projections sur $\vec{u}$ des homomorphismes de Q dans G.

1.1. *Datatypes*

RDF comme RDFS utilisent une autre sorte de termes que les urirefs et les blancs pour étiqueter les sommets d'un graphe : il s'agit des littéraux (éventuellement typés), représentés par des rectangles au lieu d'ovales dans la représentation d'un graphe. Un littéral typé est une paire constituée d'un type de données et d'une représentation d'une instance de ce type. L'interprétation d'un littéral typé est entièrement déterminée par le type de données : la fonction d'interprétation ι associe à un littéral typé la valeur associé à la représentation par le type de données (dans le cas d'un littéral simple, sans type de données, la représentation et la valeur associée sont confondues). Notons dès à présent que le domaine d'interprétation Δ contient *toutes* les valeurs contenues dans tous es types de données. La sémantique de RDFS est étendue pour prendre en compte ces types de données (par exemple, dans un modèle du triplet `(x rdf :type xsd :Boolean)`, x devra être interpété par une des deux valeurs associée à VRAI ou à FAUX dans le type de données `xsd :Boolean`).

Avec cette sémantique, la conséquence sémantique devient plus complexe à calculer. L'exemple précédent nous montre que les raisonnements doivent prendre en compte le tiers exclu (un booléen s'évalue soit à vrai, soit à faux), et il est ainsi possible d'encoder la négation. Le problème de décision DATATYPE CONSÉQUENCE SÉMANTIQUE devient ainsi au minimum Π^2P-complet. De plus, la réponse n'est plus prouvée par une application de termes à termes (mais par une arborescence de telles applications), et il n'est plus possible d'utiliser la conséquence sémantique, dans le cas général, pour définir un langage de requêtes.

1.2. *Datatypes et filtres SPARQL*

Le langages de requête SPARQL, dédié à l'interrogation de bases de données RDF/S, n'utilise pas l'extension de la sémantique RDF/S évoquée ci-dessus. Par contre, il utilise des filtres (le mot-clé FILTER) pour contraindre les réponses en imposant aux images des variables qui sont des littéraux de satisfaire certaines contraintes fonctionnelles. Ces filtres ne sont pas définis sémantiquement, leurs résultats sont le sous-ensemble des homomorphismes du pattern de la requête dans la base de faits qui satisfont aux contraintes fonctionnelles.

Le premier problème est au niveau algorithmique : si l'on cherche un ensemble de personnes qui ont le même âge, il faudra générer toutes les paires de personnes avant de supprimer celles qui ont un âge différent (au lieu de rechercher, pour chaque personne, celles qui ont le même âge). La structure imposée par ces filtres doit donc être utilisée au cours du backtrack qui calcule es homomorphismes.

Le second problème réside dans le manque d'intégration de ces filtres dans la sémantique du langage : toute extension du langage (par exemple un langage de règles construit au dessus de RDF/S) devra redéfinir la façon dont ces filtres sont utilisés pour construire le résultat de la requête.

1.3. *Objectif : une extension de RDF/S*

L'objectif de notre travail est de définir une extension de RDF/S dans laquelle les filtres (que nous appelons ici *relations calculées*) sont des objets de première classe du langage. Aux arcs usuels qui représentent les triplets RDF/S nous ajoutons des hyper-arcs qui représentent des tuples dont le premier élément désigne une valeur fonctionnelle et les arguments suivants sont es éléments de types de données. Ainsi, le graphe suivant : `{(_x rdf :type xsd :Integer), (_y rdf :type xsd :Integer), (+ _x "2" _y)}` signifie qu'il existe deux entiers x et y tels que $y = x + 2$. Ces graphes seront interprétés par une extension de la sémantique de RDF/S qui prend en compte les valeurs fonctionnelles : la fonction d'interprétation ι associe à `"2"` la valeur associée à 2, c'est à dire son codage en machine, et l'interprétation sera un modèle pour le graphe s'il existe une application ι' vérifiant, outre les contraintes de RDF ou RDFS : pour chaque relation calculée $(f x_1 \dots x_k y)$ du graphe, l'application de f sur las arguments $(\iota'(x_1), \dots, \iota'(x_k))$ retourne $\iota'(y)$.

Nous nous intéressons ici aux restrictions de ce problème général qui permettront de prouver la conséquence sémantique par une forme d'homomorphisme, afin de conserver une complexité raisonnable, de définir un langage de requêtes, et d'étendre le langage à des règles.

2. Résultats

Notre travail considère deux sémantiques différentes pour ces graphes : dans les interprétations *totales*, comme en RDF/S, toutes les valeurs possibles sont contenues dans le domaine d'interprétation Δ ; tandisque dans les interprétations *partielles*, nous n'imposons qu'aux valeurs nécessaires à l'interprétation d'être présentes. Nous étudions les problèmes liés à a satisfiabilité, à la validité et à la conséquence sémantique dans le cadre de ces deux sémantiques : alors que la première rend de nombreux problèmes indécidables, la deuxième nous autorise à garder une complexité raisonnable pour nos raisonnements.

2.1. *Un algorithme pour la conséquence sémantique*

Nous proposons l'algorithme suivant, qui calcule la conséquence sémantique de façon correcte et complète (pour la sémantique partielle) dans le cas suivant :

– le calcul de l'application des valeurs fonctionnelles sur les valeurs associées aux littéraux est décidable, et ces fonctions s'arrêtent ;

– le graphe cible est *résolvable* (*i.e.* on peut substituer à chaque sommet blanc typé par un type de données un littéral de façon à obtenir un graphe sémantiquement équivalent), ce calcul se fait par un appel à un nombre linéaire d'applications de valeurs fonctionnelles.

L'algorithme utilisé est :

1) résoudre le graphe cible G ;

2) si G est insatisfiable (calcul également linéaire dans le nombre d'application de valeurs fonctionnelles), répondre OUI, mais il peut ne pas y avoir d'homomorphisme ;

3) normaliser le graphe G (fusion de tous les somets typés par un datatype auxquels sont associés la même valeur) ;

4) calculer les homomorphismes de Q dans le graphe G' ainsi obtenu, s'il en existe un, la réponse est OUI.

Notons que nous avons modifié la définition de l'homomorphisme de façon à n'envoyer les littéraux que sur les littéraux ayant même valeur associée, et à prouver les relations fonctionnelles de la requête par le calcul de leur véracité sur les images de leurs arguments.

Si le problème de décision associé à chacune des valeurs fonctionnelles est NP-complet, alors notre problème de conséquence sémantique est également NP-complet.

2.2. *Extension aux règles*

La caractérisation de la conséquence sémantique comme un homomorphisme nous permet également de définir un langage de règles au dessus de notre extension, par la définition de règles de la forme "si hypothèse alors conclusion", où la conclusion est un graphe résolvable (en considérant que tous les blancs de l'hypothèse typés par un datatype sont des littéraux). Cette restriction nous assure que l'application d'une règle sur un graphe résolvable génère un graphe également résolvable.

Nous obtenons ainsi un mécanisme de dérivation correct et complet par rapport à la sémantique (partielle).

3. Bibliographie

Baget J.-F., « RDF Enailment as a Graph Homomorphism », *Proc. of 4th International Semantic Web Conference, ISWC 2005*, vol. 3729 of *Lecture Notes in Computer Science*, Springer, p. 82-96, 2005.

Gutiérrez C., Hurtado C. A., Mendelzon A. O., « Foundations of Semantic Web Databases », *Proc. of 23rd ACM SIGACT-SIGMODSIGART Symposium on Principles of Database Systems, PODS04*, ACM, p. 95-106, 2004.

Hayes P., RDF Semantics, W3C Recommendation, W3C, 2004.

Prudhommeaux E., Seaborne A., SPARQL Query Language for RDF, Technical report, W3C Working Draft, 2006.